Masquerade: Verifiable Multi-Party Aggregation with Secure Multiplicative Commitments

Dimitris Mouris  
ECE, University of Delaware  
jimouris@udel.edu

Nektarios Georgios Tsoutsos  
ECE, University of Delaware  
tsoutsos@udel.edu

Abstract—In crowd-sourced data aggregation, participants share their data points with curators. However, the lack of privacy guarantees may discourage participation, which motivates the need for privacy-preserving aggregation protocols. Unfortunately, existing solutions do not support public auditing without revealing the participants’ data. In real-world applications, there is a need for public verifiability (i.e., verifying the protocol correctness) while preserving the privacy of the participants’ inputs since the participants do not always trust the data curator. Likewise, public distributed ledgers (e.g., blockchains) provide public auditing but may reveal sensitive information.

We present Masquerade, a novel protocol for computing private statistics, such as sum, average, and histograms without revealing anything about participants’ data. We propose a tailored multiplicative commitment scheme to ensure the integrity of data aggregations and publish all the participants’ commitments on a ledger to provide public verifiability. We complement our methodology with two zero-knowledge proof protocols that detect potentially untrusted participants who attempt to poison the aggregation results. Thus, Masquerade ensures the validity of shared data points before being aggregated, enabling a broad range of numerical and categorical. In our experiments, we evaluate our protocol’s runtime and communication overhead using homomorphic ciphertexts and commitments for a variable number of participants.

I. INTRODUCTION

A variety of applications require gathering and aggregating data from different organizations or individuals to perform studies, collect statistics, and mine interesting patterns about the participating population. Common applications of data aggregation can be found in finance, where banks and credit unions need to gain insights on customer clusters, as well as healthcare, where aggregated data are used to discover effective treatments, and track the spread of highly infectious diseases such as coronavirus 2019 (COVID-19). In certain use cases, all participants trust the party performing the study (called data curator) with their personal data, which results in a relatively simple solution. Nevertheless, in the case of a curious curator that has incentives to peak into sensitive user data (e.g., to create targeted advertisements or even affect election results [33]), the problem becomes significantly more challenging in order to both protect the privacy of each participant and also compute the desired statistics.

Common approaches to support privacy-preserving crowd-sourcing rely on anonymizing or adding noise to the collected data so that individual inputs cannot be deduced from the output. Unfortunately, both these techniques have limitations since data anonymization can be bypassed [17], [48], while adding local noise may generate excessive accumulated noise that renders the data utility unrealistic [52]. A practical alternative for protecting the privacy of individuals in crowdsourced statistics entails adding noise to the output of the queries using differential privacy techniques so that the presence or absence of a single user cannot be detected from the query answers [22]–[24]. Unfortunately, such approaches assume a trusted curator and mostly focus on output-privacy, i.e., they do not protect the data privacy of each individual user from the curator. The notable works in [7], [25], [63] provide strong privacy guarantees and high utility, however, they still add a non-negligible amount of noise to the results. We provide thorough comparisons with related works in section VI.

In this paper, we present Masquerade, a novel construction for privacy-preserving data aggregation (PDA), a special case of secure multiparty computation (MPC) in which multiple participants (called input-parties) want to jointly compute a function over their private data while keeping those inputs private [31], [66], [67]. The threat model of MPC assumes that the client data are distributed amongst two or more computing nodes and that the data are secure as long as the computing parties do not collude. Generic MPC solutions can be used to gather statistics from multiple individuals, however, they can incur significant overheads due to their generic nature that offers support for any arbitrary computation. Additionally, many general-purpose MPC implementations, such as [19], [36], [41], [65], inherently allow only the computing parties to provide private inputs; moreover, the input parties are allowed to send any input to the aggregation, so it is still possible for a malicious participant to poison the result by uploading invalid data. To defend against such attacks, specialized checks have to be implemented in the computing nodes that require extra online communication and computation. Finally, generic MPC solutions do not provide public verifiability or auditing since data privacy relies on the computing parties not publishing their shares.

Cryptographic primitives such as homomorphic encryption allow performing meaningful calculations directly on encrypted data without decrypting it. Such schemes can be leveraged to outsource a computation to semi-trusted parties and return an encrypted result to the party that requested the computation [29], [42], [64]. In particular, homomorphic encryption can offer an elegant solution for PDA: each partic-
ipant encrypts and uploads their private data to a computing party (curator) that performs the aggregation, which in turn returns the encrypted sum to the requesting party (analyst), who can decrypt the final sum. Partially homomorphic encryption (PHE) schemes support only one mathematical operation (either addition or multiplication) to be performed very efficiently on ciphertexts, whereas fully homomorphic encryption (FHE) schemes enable both operations, but incur significant overheads especially for non-linear operations and comparisons. Nevertheless, homomorphic encryption alone does not offer any provable integrity guarantees to the public on the correctness of the computation. For instance, the public has to rely on the assumption that all computations are error-free and no participating party has introduced an invalid ciphertext to the homomorphic sum. Without special integrity checks to enable public audits, relying solely on homomorphic encryption allows computation errors to remain undetected.

Distributed ledgers (e.g., blockchains) can be used as public bulletin boards to enhance transparency and enable public verifiability. However, most of them are either public and can reveal data patterns (e.g., Bitcoin [47]), or private without support for public verifiability. The Zerocash protocol [57] and its instantiation in Zcash builds an anonymous cryptocurrency on top of Bitcoin by utilizing zk-SNARKs [6]. zk-SNARKs are non-interactive proofs that are used to hide the transaction amounts, the senders, and the receivers and still prove the integrity of the transactions. Unfortunately, zk-SNARKs require an expensive pre-processing phase to be carried out by a trusted third party to set up the system and destroy any randomness used during that step. If this randomness is leaked, any adversary can forge false proofs and break the soundness of the protocol.

**Our contribution:** This work introduces Masquerade, a novel framework for computing statistics on private data by allowing multiple users to share their data points with an oblivious curator, who is a computing party that is unable to deduce any information from the data (other than what it is learned from the finally-published result of the statistics). In particular, Masquerade involves one analyst (i.e., a party that originates the computation and will eventually reveal the statistics), \( P \) participants that share their private data, and the oblivious curator who gathers encrypted data from each participant and evaluates privacy-preserving aggregation over the data points. The participants also publish commitments to their private data to a public bulletin board, which can be used to verify the correctness of the result by any auditor. Masquerade introduces a new multiplicative-homomorphic commitment scheme that does not rely on any trusted third party.

An overview of Masquerade is illustrated in Fig. 1: The analyst outsources the computation of data aggregation to the curator and verifies the correctness of the results (we remark that Masquerade enables public audits for detecting homomorphic computation errors). The curator only learns the homomorphic ciphertexts of the participants’ messages, which do not reveal anything about their sensitive data. Similarly, the curator shares with the analyst only the encrypted result of the computation, rendering it impossible for the analyst to learn any individual information except what can be inferred from the final output of the computation. Moreover, each participant publishes commitments to their encrypted messages to a public resource that serves as an audit ledger, and the analyst combines them homomorphically to attest the correctness of the encrypted result (e.g., detect double-voting). Notably, our methodology addresses the problem of malicious participants attempting to tamper with the aggregation result, by leveraging two non-interactive zero-knowledge proof (ZKP) protocols. These protocols ensure that the participants follow the protocol faithfully, by proving to the curator that their encrypted data points lie either in a range or within a set of valid messages, without disclosing the actual plaintexts. Therefore, malicious participants cannot corrupt the homomorphic sum by sending encryptions of invalid messages.

We employ our proposed protocol to enable two classes of studies: Quantitative and Categorical. A prime example of the former class includes privacy-preserving smart metering, where users share their electricity consumption readings to a service provider over consecutive time periods to calculate their bill. Smart metering can impose a potential risk to user privacy since fine-grained readings may also disclose sensitive information about the clients’ habits and activities. Examples
in the latter class include surveys where individuals select one category among multiple valid options. For instance, a study that investigates if the juries in federal courts come from a diverse group of socioeconomic status includes sensitive personal information. This application can be instantiated natively using Masquerade, where each jury provides a private categorical input representing their ethnicity. Then, Masquerade will privately generate a histogram that represents how diverse a group of juries is. Our scheme establishes a novel way of conducting such quantitative and categorical studies by protecting the privacy of each participant.

Overall, our contributions can be summarized as follows:

- Design of a novel commitment scheme that enables homomorphic multiplication between commitments. We leverage this scheme to generate a global commitment over the encryption of the aggregation result using the individual commitments of the participants and provide verifiable guarantees to the public that there was no error in the homomorphic aggregation of the ciphertexts.
- Construct a privacy-preserving data aggregation protocol that is robust against client dropouts and malicious participants who attempt to tamper with the aggregation result.
- A data-encoding methodology to allow homomorphic aggregations for both quantitative variables (i.e, data that take numerical values) and categorical data (i.e., data grouped into discrete classes), which enables a broad range of privacy-preserving studies.

Roadmap: The rest of the paper is organized as follows: Following a preliminary discussion on the Paillier cryptosystem and commitment schemes (section II), in section III we present an overview of our protocol, our threat model, and two real-world applications. Section IV elaborates on the different components of Masquerade, i.e., our multiplicative commitment scheme and how we combine it with the Paillier cryptosystem, the non-interactive ZKP protocols that prevent malicious participants from tampering with the result of the computation, and our encoding for categorical studies. Sections V and VI evaluate the performance of our scheme and compare it with related works. Finally, our concluding remarks appear in section VII.

II. CRYPTOGRAPHY BACKGROUND

In this section, we discuss the underlying cryptographic primitives and assumptions used for developing our framework.

A. Paillier Cryptosystem

Homomorphic encryption allows performing certain mathematical operations on encrypted data that correspond to applying related operations on unencrypted data, without performing any decryptions. In this paper we employ the Paillier partially homomorphic encryption (PHE) scheme that supports homomorphic addition of encrypted messages [51]. Paillier uses a public/private key pair and supports probabilistic encryption (paired with deterministic decryption); its additive homomorphic property can be summarized as follows: Given two ciphertexts \( c_1 = \text{Enc}_{pk}(m_1) \) and \( c_2 = \text{Enc}_{pk}(m_2) \), one can compute \( c = \text{Enc}_{pk}(m_1 + m_2) \) without having to decrypt \( c_1 \) or \( c_2 \). (Here we omit the randomness for clarity.) Below we outline the basic operations of the Paillier cryptosystem.

**Key setup:** Let \( N \) be the security parameter defined as the product of two large primes \( p \) and \( q \), such that \( gcd(pq, (p-1)(q-1)) = 1 \), and \( \lambda = \text{lcm}(p-1, q-1) \). Select a random generator \( g \) in \( \mathbb{Z}_N^\ast \) and ensure that \( N \) divides the order of \( g \) by checking whether \( \mu = (L(g^\mu \bmod N^2))^{-1} \bmod N \) exists, where \( L \) is defined as \( L(u) = (u-1)/N \). The public key is \( pk = (N, g) \) and the secret key is \( sk = (\lambda, \mu) \).

**Encryption:** Randomly select a value \( \rho \) from \( \mathbb{Z}_N^\ast \) so that \( gcd(\rho, N) = 1 \). Paillier encryption is defined as a unique correspondence between \( \rho \) and plaintext \( m \in \mathbb{Z}_N \) with a value \( c \in \mathbb{Z}_N^2 \) so that \( c = \text{Enc}_{pk}(m, \rho) = g^m \cdot \rho^N \bmod N^2 \).

**Decryption:** The decryption algorithm performs an inverse mapping from \( \mathbb{Z}_N^2 \) to \( \mathbb{Z}_N \) and is defined as \( m = \text{Dec}_{sk}(c) = (L(c^\lambda \bmod N^2) \cdot \mu) \bmod N \).

**Homomorphic operations:** The Paillier cryptosystem has a notable homomorphic property that enables the product of two ciphertexts to decrypt to the sum of their corresponding plaintexts, based on the following observation:

\[
\begin{align*}
\text{Enc}_{pk}(m_1, \rho_1) \cdot \text{Enc}_{pk}(m_2, \rho_2) &= \left(g^{m_1} \cdot \rho_1^N\right) \cdot \left(g^{m_2} \cdot \rho_2^N\right) \bmod N^2 \\
&= g^{m_1+m_2} \cdot (\rho_1 \cdot \rho_2)^N \bmod N^2 \\
&= \text{Enc}_{pk}(m_1 + m_2, \rho_1 \cdot \rho_2).
\end{align*}
\]

In general, the product of \( P \) ciphertexts is equivalent to the encryption of the summation of the \( P \) corresponding plaintexts (again, we omit the randomness \( \rho \)):

\[
\prod_{i=1}^{P} \text{Enc}_{pk}(m_i) = \text{Enc}_{pk}\left(\sum_{i=1}^{P} m_i\right) \bmod N^2.
\]

Finally, the Paillier cryptosystem allows multiplication of a ciphertext with a constant, which results in the encryption of the product of the plaintext with the constant, as follows:

\[
\begin{align*}
\text{Enc}_{pk}(m_1, \rho)^{m_2} &= (g^{m_1} \cdot \rho^N)^{m_2} \bmod N^2 \\
&= g^{m_1 \cdot m_2} \cdot \rho^{N \cdot m_2} \bmod N^2 = \text{Enc}_{pk}(m_1 \cdot m_2, \rho^{m_2}).
\end{align*}
\]

B. Commitment Schemes

A secure commitment scheme is a cryptographic primitive that uses an algorithm \( C \) to enable one party, called a sender \( S^* \), to commit to a value \( x \) while keeping it hidden. To do so, \( S^* \) publishes a value \( c = C(x, r) \) that was generated using randomness \( r \); later, \( S^* \) can “open” the commitment by disclosing the used values \( x, r \). The algorithm \( C \) binds \( S^* \) to the hidden value \( x \) and offers provable guarantees to a receiving party \( R^* \) that the value revealed later was actually the same to the one \( S^* \) originally committed to when \( c \) was first published. Early commitment schemes proved that it is possible for example to have two parties play a card game or flip a coin via the telephone without having to trust each other [8], [12], [62].
For any commitment scheme to be secure, it must satisfy two basic properties: binding and hiding. The first property guarantees that upon committing to a secret value \( x \), \( S^* \) cannot change \( x \) later. Formally, for all non-uniform probabilistic polynomial-time algorithms that output \((x_1, r_1)\) and \((x_2, r_2)\), the probability that \( C(x_1, r_1) = C(x_2, r_2) \) with \( x_1 \neq x_2 \), is negligible. The second property requires that the commitment \( c \) should not reveal any information about the value \( x \) before opening, that is, for all non-uniform probabilistic polynomial-time algorithms, the probability of extracting any information about \( x \) from \( c \) should be negligible. A more recent commitment scheme, named after T. P. Pedersen, features an additive homomorphic property so that for messages \( x_1 \) and \( x_2 \) with blinding factors \( r_1 \) and \( r_2 \) we have: \( C(x_1, r_1) \cdot C(x_2, r_2) = C(x_1 + x_2, r_1 + r_2) \) \cite{54}. In this work, we define a new multiplicative commitment scheme (discussed in section IV-A) that enables computational integrity for Masquerade.

C. Zero-knowledge Proofs (ZKPs)

A ZKP is a cryptographic protocol that allows a prover \( P \) to assert the truth of a statement to a verifier \( V \) without revealing any further information \cite{32}. State-of-the-art systems that use non-interactive ZKPs (e.g., \cite{9}) as well as Succinct Non-interactive/Transparent Arguments of Knowledge – SNARKs/STARKs (e.g., \cite{5, 6, 28, 44–46, 53}) focus on proving general-purpose statements and may incur high overheads compared to bespoke solutions. At a high-level, ZKPs must satisfy the following properties:

- **Correctness:** If \( P \) is honest, \( V \) will accept the proof.
- **Soundness:** If \( V \) is honest, no cheating \( P \) can convince \( V \) to accept a false statement, except with negligible probability.
- **Zero-knowledge:** If both \( P \) and \( V \) are honest, then the proof does not reveal anything about \( P \)'s private statement, except the fact that it is true.

D. Fiat-Shamir Heuristic

Fiat and Shamir introduced a method (known as the Fiat-Shamir heuristic) that eliminates the interaction in public coin proof-of-knowledge protocols and make them non-interactive \cite{26}. Essentially, the Fiat-Shamir heuristic relies on the random oracle model \cite{4} in order to substitute the random challenge generated by the verifier with a challenge that is the output of a hash function over the previous protocol messages. The sequence of messages (and most importantly their commitments) in the protocol are referred to as a transcript. Modeling the hash function as a random oracle prevents the prover from guessing its output before the commitments are generated.

### III. Our Problem Statement

#### A. Overview

Gathering and aggregating data from multiple individuals and/or organizations enables powerful analytics. For instance, companies can collect meaningful information about their clients’ behavior and demographics, while healthcare researchers can discover more effective treatment methods to accelerate the overall healthcare facility. Such studies correspond to one of two categories based on the type of data being used: The first class includes variables that represent quantities (i.e., quantitative data) that take numerical values (such as age, price, and profits). The variables in the second class can take one value from a fixed set of valid options, and therefore are called categorical data (such as age-group, nationality, and employment status). In the case of a quantitative study, the curator applies a function \( f \) directly to the participants’ data (e.g., compute the average age of consumers or a weighted arithmetic mean based on electricity cost), while in a categorical study the curator applies \( f \) in each different category to generate a histogram (e.g., find the total number of consumers by nationality). The latter can be depicted as a bar graph that shows the frequency distribution of the participants’ responses.

The problem of Private Data Aggregation (PDA) aims to compute similar aggregations and frequencies while tolerating a curious curator that has incentives to peek at sensitive data points (e.g., to enable targeted advertisements). PDA enables many useful applications by assuring the participants that their data will only be used towards the agreed analysis and private data cannot be tracked back to each individual; notably, the curator will never access the sensitive data. In this work, we are interested in defining a secure PDA protocol that can be used to preserve individual privacy for both quantitative and categorical studies. To further motivate our case study, we summarize an example from each of the two scenarios.

**Categorical analysis:** This class enables individuals to anonymously take part in surveys where they choose their replies from fixed sets of answers. For example, university course evaluations include questions with categorical data, where each response is a one-hot encoding from a set of possible answers (e.g., “The instructor used the class time effectively,”, with four possible responses a) Strongly Disagree, b) Disagree, c) Agree, d) Strongly Agree on a Likert scale \cite{39}). These evaluations comprise an essential indicator for the faculty and their departments to get feedback and improve their classes. In course evaluations, it is crucial that there is no link between the participant and their responses (anonymity requirement) so faculty are not biased towards individuals based on their responses, and most importantly, individuals are more likely to provide more accurate comments. In this case, public verifiability guarantees to the students that their evaluations were counted and also the students can prove that they only submitted one response.

**Quantitative analysis:** In this category, individuals can anonymously report numerical values that will be used to compute a total sum, an average, or a weighted average. A notable
example is a smart-metering application which can be applied to multiple participating households. Smart meters communicate the electrical usage almost in real-time to provide better system monitoring and customer billing than traditional meters. However, there are inherent privacy concerns since fine-grained measurements (e.g., one every 15 minutes) may reveal personal information about the number of people in a household and their activity patterns [14], [20]. Masquerade helps solve this problem by computing aggregated data over multiple participating households for a given period that would still be sufficient for smart grid operators to perform enhanced system monitoring and optimize prices, but at the same time will conceal private information. In smart metering, public verifiability is essential to convince all the households that the result was computed correctly. Finally, Masquerade naturally extends this application by computing the weighted arithmetic mean of the participating households, as electricity rates can vary based in the area. This allows participants and communities to compare their private electricity consumption based on their (public) area price and gain useful insights about the applicable electricity rates.

B. Threat Model

The different applications supported by Masquerade require protection against a variety of threats. Our threat model assumes malicious participants, an honest-but-curious analyst, and a more elaborate mixed model for the curator that comprises the passive-corrupt (i.e., semi-honest or honest-but-curious) and the fail-corrupt models [27]. This section elaborates on the specific assumptions for each role in our protocol.

Curator: A passive-corrupt or semi-honest curator may have incentives to peek into the participants’ sensitive data in order to extract information about the encrypted messages. In our approach, participants’ inputs are protected using a secure probabilistic PHE scheme (our protocol relies on the Paillier encryption scheme with a 2048-bit modulus $N$). A fail-corrupt curator may stop the communication from the participants at an arbitrary time during the protocol with a result of altering the final sum (e.g., by being offline for a certain period of time and failing to aggregate all the ciphertexts). Additionally, under the fail-corrupt model, the curator may introduce non-adversarial or random faults, which we assume could affect the integrity of the computation [58]. In this paper, we borrow the mixed model introduced in [27], in which the adversary may passively-corrupt and/or fail-corrupt the curator. As PHE does not offer any native integrity protections, we introduce a novel multiplicative homomorphic commitment scheme (described in section IV-A) that provides public verifiability without compromising the participants’ privacy. The analyst can also employ these public commitments to verify the computational integrity of the homomorphic result.

Participants: We assume both honest and malicious participants. Participants in the former category will follow the protocol and are solely interested in learning the aggregation result. On the other hand, malicious participants may attempt to tamper with the analysis by encrypting invalid inputs (e.g., a large negative number). To address this threat, we introduce two non-interactive ZKP protocols that allow the participants to prove to the curator that the provided data encrypt a valid message. Additionally, to defend against Sybil attacks that create a large number of illicit clients in order to disproportionately influence the result [21], both the data curator and the analyst agree on a list of identities from which they will accept inputs. Active-adversary attacks can be further thwarted using a Public-Key Infrastructure that issues unique certificates to participants and enables them to sign their messages; in this case, all participants in our protocol can verify the message signatures to prevent forgeries [1]. As soon as the private aggregation is finished, the curator sends to the analyst the list of participants whose zero-knowledge proof and commitment was verified successfully to include their commitments for the verification of the encrypted sum. Use of ZKP verification ensures that our protocol remains robust even if some of the participants choose to submit malicious inputs, or even if they choose not to participate after they have registered, as invalid entries are detected before aggregation and are safely ignored.

Analyst: Our threat model assumes a passive-correct analyst that is interested in learning the aggregation results but also has incentives to extract information about individuals’ data. In our protocol, the analyst learns only the encrypted sum, and nothing is revealed about each participant’s ciphertext (the analyst can only learn the aggregation output). Here we focus on building a privacy-preserving data aggregation protocol that is robust against malicious participants, so we consider the bulletin board as an append-only ledger available online. Indeed, this bulletin board should be auditable by anyone (e.g., public auditors) and can be maintained by the analyst using standard consensus methods [50], or using a public blockchain network (e.g., Ethereum).

Collusion attacks: Our threat model is aligned to the models used in secure two-party computation protocols and it assumes that the curator and the analyst will not collude. Instantiating our protocol assuming two parties with mutual distrust enables computing the aggregation without exposing sensitive information about the participants’ data [35], [66].

IV. PRIVATE DATA AGGREGATION PROTOCOL

In this section, we describe Masquerade, our cryptographic construction for secure PDA. Our protocol supports quantitative data by design, and we extend our construction to support categorical data, as discussed in section IV-E. Each participant (a) locally encrypts their private data using Paillier, (b) publishes in a bulletin board a multiplicative commitment on the ciphertext, (c) generates a non-interactive ZKP that enables its recipient to verify that the ciphertext is the encryption of a message that lies in a range of valid messages, and (d) sends their encrypted data along with a proof to the curator. The curator first verifies the validity of the ZKP and the correctness of the commitment, and then homomorphically
During the reveal phase, both the secret message and the public exponent privately revealed. The analyst audits the protocol by accumulating the homomorphic commitments from the ledger and verifying that they open with the encrypted sum. Since the commitments and the final sum are public, any participant can repeat the same process and be convinced that their private data were included in the computation, as well as verify the correctness of the encrypted aggregation. We remark that the need for public verifiability remains critical in real-world applications with potentially untrusted participants. Therefore, Masquerade explicitly enables the participants to verify that no errors were introduced in the aggregation result (e.g., by omitted inputs or due to other random faults). Upon verifying that the curator’s result is correct, the analyst decrypts and publishes the result along with a proof of correct decryption of the final result.

**Benefits:** An essential property of our scheme is that it does not require any interactive communication between the participants and the curator. Each participant generates the encrypted data, the ZKPs, and the commitments locally and then can disconnect from the protocol after submitting them. Since our commitment scheme does not reveal anything about the participants’ data, any public bulletin board or ledger that does not rely on a trusted setup would be sufficient.

Furthermore, if some clients deliberately stop participating in the protocol (i.e., participant dropout), our scheme remains resilient since we do not rely on any user secret for decryption of the final result. In contrast, in earlier PDA protocols that are based on secret-sharing, a single client can easily corrupt the protocol by just withdrawing their participation. We discuss and compare with related works in section VI.

**A. Our Multiplicative Commitment Scheme**

Recall from section II-B that the Pedersen commitment scheme has additive homomorphism [54]. Here we propose a new commitment scheme based on RSA [56] with multiplicative homomorphism. Let $N$ be a public RSA modulus so that $N = p \cdot q$, where $p, q$ are two secret primes, and let $e$ be a public prime number that satisfies $e > N^2$ and $gcd(e, \phi(N^2)) = 1$. Moreover, let $m_0$ be a public element of maximum order in group $\mathbb{Z}_{N^2}^*$ [43, Alg. 4.83]. To bind to a secret message $m \in \mathbb{Z}_{N^2}$, the sender $S^*$ generates a random secret $r \in \mathbb{Z}_{N^2}^*$ and calculates $c$ as follows:

$$c = \mathcal{C}(m, r) = m^e \cdot g_m^{r} \mod N^2.$$  \hfill (4)

Assuming that the RSA problem with modulus $N^2$ and public exponent $e$ is hard to invert, this commitment scheme (intuitively) satisfies the hiding and binding requirements. During the reveal phase, both the secret message $m$ and the randomness $r$ are published, and the verifier checks if they produce the same commitment $c'$ as $c$.

**Theorem 1.** The commitment scheme with multiplicative homomorphism from Eq. 4 satisfies the hiding and binding properties.

**Proof.** First, we show that the scheme is perfectly hiding. Sender $S^*$ confirms that $N^2 < e$ and $e$ is a prime, so that $gcd(e, \phi(N^2)) = 1$ and $e$ is invertible mod $\phi(N^2)$. We will show that $c = m^e \cdot g_m^{r} \mod N^2$ is indistinguishable from random values: Since $g_m$ is an element of maximum order in $\mathbb{Z}_{N^2}^*$ (by construction) and $r$ is chosen uniformly at random in $\mathbb{Z}_{N^2}^*$, then $g_m^r \mod N^2$ is also uniformly distributed in $\mathbb{Z}_{N^2}^*$. Therefore, the product $m^e \cdot g_m^{r} \mod N^2$ is also uniformly distributed in $\mathbb{Z}_{N^2}^*$ and indistinguishable from a random value, and therefore, nothing can be learned about $m$ from $c$.

Next, we prove that our scheme is computationally binding under the RSA assumption (i.e., the problem of computing $e^{th}$ roots modulo a composite $N^2$ is computationally hard). Our goal is to show that if a sender $S^*$ can find two different messages $m, m'$ with corresponding randomness $r, r'$ that collide to the same commitment $c = c'$, this would imply we can easily compute $e^{th}$ roots modulo $N^2$ so that the RSA problem would be easy (contradiction). To show this, we assume it is possible to find $m \neq m'$ and $r \neq r'$, so that $c = m^e \cdot g_m^{r} = c' = m'^e \cdot g_m^{r'} \mod N^2$. In this case, $m^e \cdot g_m^{r} = m'^e \cdot g_m^{r'} \mod N^2 \iff g_m^{r-r'} = (m'^m)^e \mod N^2$.

Using the fact that $r, r' \in \mathbb{Z}_{N^2}$ and $e$ is a prime larger than $N^2$, it holds that $r - r' < e$ and also $gcd(r - r', e) = 1$. Thus, from Bézout’s identity, there exist integers $A, B$ (that can be computed in polynomial time using the extended Euclidean algorithm), so that $A \cdot (r - r') + B \cdot e = gcd(r - r', e) = 1$. Then, we have:

$$g_m^{1/e} = g_m^{A(r-r') + B e} = (g_m^{r-r'})^A \cdot g_m^{B e} \mod N^2 = ((m^e/m)^e)^A \cdot g_m^{B e} \mod N^2 = ((m^e/m)^A) \cdot g_m^{B e} \mod N^2,$$

and thus, $g_m^{1/e} = (m^e/m)^A \cdot g_m^{B e} \mod N^2$.

The above steps in Eq. 5 can efficiently compute the $e^{th}$ root of $g_m$ modulo $N^2$, which contradicts the hardness assumption of RSA for a sufficiently large composite $N^2$. Therefore, the original assumption that $S^*$ can bind a commitment $c$ to more than one message is false, as expected.

From Eq. 4 we observe that our commitment has a multiplicative homomorphic property; this means we can compute a commitment for the multiplication of two messages $m_1$ and $m_2$ by knowing the individual commitments to these two messages. To open this new commitment, the randomness $r_1$ and $r_2$ used for each individual commitment should be added. Note that this is distributed identically to a fresh commitment to $m_1 \cdot m_2$. More formally:

$$\mathcal{C}(m_1, r_1) \cdot \mathcal{C}(m_2, r_2) \mod N^2 = (m_1^e \cdot g_m^{r_1}) \cdot (m_2^e \cdot g_m^{r_2}) \mod N^2 = ((m_1 \cdot m_2)^e \cdot g_m^{r_1 + r_2}) \mod N^2 = \mathcal{C}(m_1 \cdot m_2, r_1 + r_2).$$  \hfill (6)
Eq. 6 can be extended to $P$ messages as follows:

$$\prod_{i=1}^{P} C(m_i, r_i) = C \left( \prod_{i=1}^{P} m_i, \sum_{i=1}^{P} r_i \right) \mod N^2 \quad (7)$$

In the next section, we describe how the analyst leverages this property to generate a commitment over the encrypted sum without having access to the individual encrypted messages, but only to the public commitments of each encrypted message. By the hiding property (Theorem 1), each commitment to a message does not reveal anything to the analyst about the message itself. Additionally, the binding property guarantees the computational integrity of ciphertext aggregation, as an error in the aggregation result would prevent an auditor from successfully opening the aggregate commitment. Masquerade offers public verifiability by publishing all the commitments to a ledger and enabling any party to audit them.

B. Homomorphic Commitments on Homomorphic Data

We now discuss our novel private data aggregation scheme, depicted in Fig. 2. First, the analyst generates a Paillier key-pair $sk, pk$ as well as the random prime $e$ and the maximum-order element $g$ required for our commitment scheme (section IV-A) and publishes Paillier’s public key $pk$ (including modulus $N^2$), $e$ and $g_m$. Notably, we use the same $N^2$ both as the public commitment modulus and as Paillier’s encryption modulus. Then, each participant probabilistically encrypts their private message using Paillier and commits to the generated ciphertext (recall, these commitments do not reveal anything about either the ciphertext or the plaintext). Then, the participant sends their Paillier ciphertext to the curator along with the randomness used for the commitment and publishes the commitment value to the ledger. (We omit ZKPs for now.) The curator calculates the homomorphic addition of all the participants’ messages, adds the individual commitment randomness $r_i$, and publishes the Paillier ciphertext along with the sum of all $r_i$’s to the bulletin board. Finally, the analyst verifies the integrity of the homomorphic result using the formula of Eq. 7 to ensure the multiplication of all individual client commitments along with the aggregated randomness equals the commitment over the encrypted sum.

For example, given two participant messages $m_1$ and $m_2$, the curator receives two ciphertexts $Enc_pk(m_1)$ and $Enc_pk(m_2)$ and computes $Enc_pk(m_1) \cdot Enc_pk(m_2) \mod N^2$ (here, we have omitted the randomness inputs of Paillier encryption). Moreover, the curator sums the commitments’ randomness ($r_1+r_2$). The analyst reads the published commitments to these two ciphertexts and can verify the commitment to the encrypted sum without having access to the individual ciphertexts as follows:

$$C(Enc_pk(m_1), r_1) \cdot C(Enc_pk(m_2), r_2) \mod N^2$$

$$= (Enc_pk(m_1)^e \cdot g_m^r_1) \cdot (Enc_pk(m_2)^e \cdot g_m^r_2) \mod N^2$$

$$= (Enc_pk(m_1) \cdot Enc_pk(m_2))^e \cdot (g_m^{r_1+r_2}) \mod N^2 \quad (8)$$

$$= C(Enc_pk(m_1), Enc_pk(m_2) \mod N^2, r_1 + r_2) = C(Enc_pk(m_1 + m_2), r_1 + r_2).$$

Therefore, an auditor can compute and open the commitment for the encrypted sum that they received from the curator by using the accumulated randomness. In our implementation, by default the analyst plays the role of an auditor but we also enable any party to further verify the protocol by inspecting the public ledger. With a successful commitment opening, an auditor is assured that no errors occurred during the aggregation.

Additionally, we extend this core protocol to support computing the weighted arithmetic mean based on both the participants’ private inputs and the public weights. From Eq. 3, it follows that the curator can multiply encrypted participant inputs with constant weight values $w_i$, where $i \in [1, P]$. Based on the previous example, for ciphertexts $Enc_pk(m_1)$ and $Enc_pk(m_2)$, the curator computes $Enc_pk(m_1)^{w_1} \cdot Enc_pk(m_2)^{w_2} \mod N^2 = Enc_pk(w_1 \cdot m_1) \cdot Enc_pk(w_2 \cdot m_2) \mod N^2$, and also sums the commitments’ randomness. Finally, the analyst incorporates the weights in the commitments received from the participants by applying the following formula: $w_i^e \cdot C(Enc_pk(m_i), r_i) = (w_i \cdot Enc_pk(m_i))^e \cdot g_m^r_i = C(Enc_pk(w_i \cdot m_i), r_i).$

C. Public Verifiability for Aggregator

Masquerade enables the participants to audit not only the curator, but also the analyst and efficiently verify that the published sum is the correct decryption of the ciphertext that the curator published to the ledger. We adopt the ZKP protocol for proving ciphertext equality over $N^{s+1}$ modulus with $s = 1$ from [18], and the Fiat-Shamir heuristic in the random oracle model to convert the ZKP to a non-interactive variant. Let $Pai_{sum}$ be the encrypted result and $sum$ be the decryption.
publised by the analyst. We further provide public verifiability that the analyst decrypted and published the correct result: The analyst first computes encrypts sum using randomness ρ into Pai′sum. Consecutively, the analyst computes a ciphertext Pai0, which is the homomorphic result of Pai′sum minus Pai′sum, and posts Pai0, Pai′sum, along with the randomness ρ to the ledger. By construction, if Pai′sum is an encryption of sum, then Pai0 is an encryption of zero. Finally, the analyst uses a non-interactive ZKP (Appendix A, Protocol 1) and provides public verifiability that Pai0 is an encryption of zero, which means that Pai′sum is an encryption of sum. Notably, it is not possible for the analyst to cheat and publish a wrong sum, as Pai0 would not be an encryption of zero and this would break the soundness of the ZKP [18].

D. Protecting Against Malicious Clients

As participants contribute their data encrypted, the curator has no way to detect if a malicious client performs a data pollution attack by sending an encryption of an overwhelmingly large number. Masquerade bounds the influence of each participant in the aggregate results by restricting the values they can input into the protocol. Specifically, to mitigate such attacks, we incorporate two non-interactive ZKP protocols, namely a range proof, and a set-membership proof. The former enables the participants to prove to their Paillier ciphertext encrypts a message m that lies within a valid range, i.e., m ∈ {0, 1, ..., ℓ} [11], [40], while the latter proves that m lies within a set of messages, i.e., m ∈ {m1, m2, ..., mk} [3], [18]. In both cases, the client manages to convince the data curator by only showing the Paillier encryption of m without leaking any additional information about m.

As discussed in Section II-C, ZKP protocols must satisfy the three basic properties, completeness, soundness, and zero-knowledge. Completeness guarantees that an honest participant (P) can always convince the curator (V) about a valid statement. Soundness guarantees that if the curator is honest and the participant’s private data are not well-formed (i.e., are not within a pre-specified range or a set of public messages), then the curator will reject the proof with overwhelming probability.

We formalize the soundness by introducing a security parameter t: the probability of an adversary convincing an honest verifier in range proofs is 2−t, and in set-membership proofs it is A−t, where A is the size of the randomness used for the protocol. Finally, zero-knowledge guarantees that if both the participant and the curator are honest, then the proof does not reveal anything about the private participant’s data, except that they are well-formed.

We utilize the zero-knowledge range proof protocol for quantitative aggregations, while the set-membership proof is used to assert that the participants select a message from some predefined categories (described in section IV-E). We instantiate these two ZKP protocols as public coins in the random oracle model [4] via the Fiat-Shamir heuristic (discussed in Section II-D) to eliminate the interaction and transform the interactive proofs to their non-interactive variants [26]. Therefore, the data curator verifies the correctness of the proof for
whereas the prover and verifier algorithms are shown as \( V \) and \( \mathcal{V} \), respectively.

In Fig. 2, the proof for the participant \( i \) increments the sum if and only if the proof was accepted. The ciphertext sent by each participant and homomorphically each category and end up with individual accumulators. Homomorphic addition of ciphertexts will result into summing the bits from the categories and increment that counter (e.g., "adult"). Each participant could contribute an encryption of a binary number. Unfortunately, in cases that the response should be "one-hot encoded" (such as the "age group"), this solution is not sufficient since it would enable malicious participants to contribute to more than one category and tamper with the aggregation result. Although in each invocation they could submit a valid encryption and a proof, the set of their responses could be invalid (e.g., submit multiple encryptions of True).

Therefore, Masquerade introduces an encoding that enforces the participants to select 1-out-of-\( K \) categories using a single message that encodes both the selected category as well as the remaining \( K - 1 \) non-selected categories. We encode each category as a power of \( 2^{\frac{N}{K}} \), where \( N \) are the bits of the plaintext: specifically, the first category is represented as \( (2^{\frac{N}{K}})^0 = 1 \), the second category is represented as \( (2^{\frac{N}{K}})^1 = 2^{\frac{N}{K}} \), and so on. In Fig. 4(a) we illustrate these encodings. We remark that performing homomorphic addition between ciphertexts from the same category will increment the \( N/K \)-bit counter for the specific category, and leave the counters for the \( K - 1 \) remaining categories unmodified. The accumulations for each selected attribute create a one-dimensional histogram with \( K \) categories, as illustrated in Fig. 4(b). We complement our protocol with the non-interactive set-membership proof described in section IV-D to guarantee benevolent client behavior; in this case, each participant encrypts one of \( K \) possible messages (i.e., powers of \( 2^{\frac{N}{K}} \)) and creates a zero-knowledge set membership proof that the ciphertext is the encryption of one of these powers.

Using the same number of plaintext bits, our protocol also supports responses that enable the participants to select multiple categories at the same time, if those are valid answers. To support such messages, the get-membership proof should include all the possible combinations of valid responses. For instance, a response that includes both categories \( C_1 \) and \( C_3 \) from Fig. 4(a) would be in the form \( 00\ldots00 | \ldots | 00\ldots01 | 00\ldots00 | 00\ldots01 \). Adding the above response with another response for category \( C_1 \) will result in \( 00\ldots00 | \ldots | 00\ldots01 | 00\ldots00 | 00\ldots01 | 00\ldots01 | 00\ldots00 | 00\ldots10 \), which accumulates both responses.

Notably, having large plaintext sizes and attributes over a certain number of categories, our encoding technique can also support multidimensional histograms. For instance, extending the example of age groups, the analyst can also consider if a participant is a smartphone user. We use these two variables (i.e., "age groups" and "smartphone user") to form a two-dimensional array with all possible combinations, and encode each index in that array into a unique category from Fig. 4(a).

Taking these two requirements into account, we design a specialized encoding to incorporate different categories into one encrypted message. A possible solution for categorical variables would be to use multiple invocations of our quantitative PDA protocol so that every invocation would aggregate one category. Each participant could contribute an encryption of either a "one" or a "zero", depending on their selection, and also prove in zero-knowledge that the submitted ciphertext is an encryption of a binary number. Unfortunately, in cases that the response should be "one-hot encoded" (such as the "age group"), this solution is not sufficient since it would enable malicious participants to contribute to more than one category and tamper with the aggregation result. Although in each invocation they could submit a valid encryption and a proof, the set of their responses could be invalid (e.g., submit multiple encryptions of True).

V. EXPERIMENTAL EVALUATIONS

The goal of our experimental evaluation is to assess the performance of our protocol and understand how it scales
with an increasing number of participants. On a theoretical level, we expect Masquerade to scale linearly to the number of clients and the two zero-knowledge proof protocols to dominate the execution time, especially as the soundness parameter \( t \) increases. Finally, the set-membership proof is more computationally intensive than the range proof and therefore we expect the quantitative studies to be faster than categorical.

### A. Experimental Setup

For our experimental evaluation of the curator and the analyst, we used two t3.2xlarge AWS EC2 instances running with eight virtual processors up to 2.5 GHz and 32 GB RAM; both hosts used Ubuntu 20.04 with Linux kernel 5.4. For the experiments evaluating the participants, we used a Lenovo laptop with an i7-8650U CPU running at 1.90 GHz, and we have implemented our Masquerade framework in Go 1.16.\(^1\) In our experiments, the latency between the curator and the analyst was negligible (about 0.014 ms) as both are hosted in AWS instances. The communication between the participants and the AWS instances (curator and analyst) is 9.65 ms on average. The back-end of the curator is fully parallelizable and can take advantage of all the available threads of the host machine. For every participant, the curator spawns a new thread to verify the zero-knowledge proof of the participant and aggregates their private data only if the ZKP verification was successful. This optimization provides a speed-up proportional (roughly) to the number of parallel cores of the curator’s host. Without loss of generality, in our implementation the analyst maintains a public append-only ledger. The ledger can also be maintained by any other trusted third-party, or be distributed between the analyst and the curator. Moreover, Masquerade supports ledgers deployed on a public blockchain: our implementation offers end-to-end support for publishing commitments on Ethereum using the Ganache-CLI, and was evaluated on the Ropsten testnet. Finally, both the Paillier cryptosystem and our commitment scheme are instantiated with a 4096-bit modulus \( N^2 \) following the key size recommendations of [2].

### B. Performance Evaluation

In Fig. 5 we present the online protocol costs of Masquerade assuming both honest and potentially malicious participants. In the former – ideal – case, the participants’ ciphertexts are well-formed and the curator accepts all the encrypted data directly, eliminating the need for creating and verifying ZKPs. We use the timings of this simplified protocol as a baseline to demonstrate the trade-off of the zero-knowledge protocols for the quantitative and categorical studies (note, the honest participant cost does not depend on the type of study). Fig. 5 demonstrates the experimental timings for quantitative variables (i.e., green triangles), categorical variables (i.e., red squares), a baseline for honest participants (i.e., blue diamonds), and the timing for an auditor (i.e., yellow circles) with an increasing number of participants and soundness parameter \( t = 60 \) bits.

The experimental timings include the computation performed by the analyst and the curator until the result of the aggregation is published by the analyst. We note that Fig. 5 does not include the offline cost for key-generation or participant cost (these are discussed later in this Section) since all the participants can pre-compute their ciphertexts, proofs, and commitments before engaging in the protocol. Finally, the timings also include the latencies, which have negligible performance cost compared to the ZK-related operations. For the reported cost of the categorical study in Fig. 5 we used a set with \( K = 4 \) classes. As expected, for both types of studies we observe a linear increase to the total aggregation cost as the number of participants grows. Additionally, for up to 8 participants the cost of the quantitative and categorical remains somewhat constant, due to the thread parallelization in the curator’s back-end. When we compare both trends that use ZKPs with the trend of “Honest Participants”, we observe that protecting against malicious participants incurs about two orders of magnitude additional overhead. Likewise, the quantitative study (i.e., green triangles) is about half an order of magnitude faster than the categorical study (i.e., red squares), which is attributed to the complexity of Protocol 3. Finally, we observe that the time to audit the ledger is almost constant with an increasing number of participants as it mostly comprises modular multiplications. In the following paragraphs, we analyze the individual performance of the participants, the curator, the analyst, and the auditor and discuss how each one of them affects the total performance of our protocol.

#### Participant Perspective:

The computation cost of each participant includes the Paillier encryption cost, the ZKP cost, and the commitment cost. The range proof is much more computationally intensive than the set-membership proof. The range proof is also more complex, with the introduction of the set-membership proof as an additional layer of security. This results in a higher computation cost for the participants.

### Table I: Size of the Zero-knowledge Proof Protocols

<table>
<thead>
<tr>
<th>Soundness Parameter (( t ))</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>120</th>
<th>140</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Range Proof (KB)</strong></td>
<td>36.0</td>
<td>57.4</td>
<td>78.1</td>
<td>103.1</td>
<td>118.4</td>
<td>142.2</td>
</tr>
<tr>
<td><strong>Set Proof (KB)</strong></td>
<td>309.6</td>
<td>464.5</td>
<td>619.3</td>
<td>774.1</td>
<td>929.0</td>
<td>1083.9</td>
</tr>
</tbody>
</table>

\(^1\)Our code will be available on GitHub after the blind-review.
and the commitment cost. Our protocol encourages the participants to pre-compute everything offline. Both the Paillier encryption cost and the cost of committing to encrypted data are negligible compared to the ZKP generation. Fig. 6 shows the proof generation time for participants for both range and set-membership proofs with varying soundness parameters $t$. Similarly, Table I shows the ZKP size in KBytes with an increasing $t$ for range and set-membership proofs.

Both the computation and the space costs of the range proofs scale linearly to the soundness parameter $t$. Similarly, the computation and the space costs of the set-membership proofs scale linearly to $t$, but are also affected by the number $k$ of messages in the set. In Table II we show the timings for the prover and verifier for the set-membership proof with increasing $t$ for range and set-membership proofs.

<table>
<thead>
<tr>
<th>Set Size ($K$)</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prover Time (Lenovo) (sec.)</td>
<td>1.92</td>
<td>2.94</td>
<td>7.92</td>
<td>14.61</td>
<td>30.55</td>
<td>63.83</td>
</tr>
<tr>
<td>Verifier Time (EC2) (sec.)</td>
<td>2.08</td>
<td>2.61</td>
<td>6.97</td>
<td>13.26</td>
<td>31.93</td>
<td>65.14</td>
</tr>
</tbody>
</table>

Fig. 6. Zero-Knowledge Proofs Performance. Time measurements in seconds for both range and set-membership proofs for the prover and the verifier with an increasing soundness parameter $t$ in bits and $K = 4$.

In our implementation of Masquerade, the analyst maintains the public ledger. Thus, the communication cost of the analyst entails receiving the individual commitments from each participant, as well as receive a list of identifiers from the curator corresponding to the participants whose ZKPs were correct. Performance-wise, in addition to decrypting the encrypted sum, the only additional cost of the analyst is to support public verifiability by aggregating all participant commitments and showing that it correctly opens with the homomorphic sum (Alg. 4 in Fig. 3).

Auditor Perspective: Finally, any auditor can access a copy of the ledger in order to verify that the protocol was executed correctly, which allows public verifiability. The auditor uses the accumulated randomness and verifies that the participants’ homomorphic commitments open with the encrypted sum. This verification is similar to the one that the analyst performs in Fig. 3 Alg. (4), line 2. Lastly, as mentioned in section IV-B, the auditor verifies that the published decrypted sum is indeed a correct decryption of the ciphertext that the curator published to the ledger. We measured the auditor timings in Fig. 5 (i.e., yellow circles) with modern Lenovo laptop (the same machine was also used for the participants) and we observe almost constant overhead.

VI. RELATED WORK

In this section, we discuss several recent works in PDA that rely on cryptographic techniques such as homomorphic encryption, secure multi-party computation, and differential privacy. All these existing solutions introduce different trade-offs between computation and communication costs, depending on the underlying privacy technique they employ. Interestingly, some protocols provide robustness against dropouts,
A comparison with existing PDA schemes based on the cryptographic technique they utilize, the type of variables they support and their robustness against dropping participants and malicious inputs.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Cryptographic Technique</th>
<th>Quant. Vars</th>
<th>Categ. Vars</th>
<th>Robust Against Dropouts</th>
<th>Public Verifiability</th>
<th>Robust Against Malicious Inputs</th>
<th>Proof Scaling Quant./Categ.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shi et al. [63]</td>
<td>THE &amp; Diff. Privacy</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>N/A</td>
</tr>
<tr>
<td>Chan et al. [15]</td>
<td>THE &amp; Diff. Privacy</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Joyce et al. [34]</td>
<td>THE</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Danezis et al. [20]</td>
<td>THE</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Leontiadis et al. [37]</td>
<td>THE</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Leontiadis et al. (PUDA) [38]</td>
<td>THE</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Bonawitz et al. [10]</td>
<td>Pairwise Masking</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Burkhart et al. (SEPIA) [13]</td>
<td>Secret-Shar., Generic MPC</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>Quadratic / N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Giannopoulos et al. [30]</td>
<td>Secret-Shar., Generic MPC</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Narula et al. (zkLedger) [49]</td>
<td>Pedersen Commitments</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>Quadratic / N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>Corrigan-Gibbs et al. (Prio) [16]</td>
<td>Secret-Shar.</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>Linear / Linear</td>
<td>Linear / Linear</td>
</tr>
<tr>
<td>This Work (Masquerade)</td>
<td>PHE &amp; Commitments</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>⬗</td>
<td>Constant / Linear</td>
<td>Linear / Linear</td>
</tr>
</tbody>
</table>

1 zkLedger is the only related work that achieves both public verifiability and being robust against malicious inputs, however, the transaction verification is quadratic to the number of participants causing major scalability issues (the authors only experiment with at most 20 participants). Additionally, zkLedger does not support categorical variables.

2 Prio introduces a mechanism called affine-aggregatable encodings (“AFEcs”) to encode a series of bits as quantitative or categorical variables. Prio does not guarantee public verifiability and each client only communicates with the servers (no support for a distributed ledger).

3 Threshold Homomorphic Encryption (THE): The private key is shared between $\mathcal{O}$ parties, where each bit is verified by a different SNIP, and computes

```
while only a few are secure against malicious clients that provide malformed inputs. Moreover, Masquerade and only two other works allow the participants to verify the outcome of the protocol. In the following paragraphs, we discuss these existing works and how they compare with Masquerade.

In Secure Multi-Party Computation (MPC), multiple input-parties want to jointly compute a function over their private data while keeping those inputs private [31], [66], [67]. Most common MPC constructions are either based on Yao’s garbled circuits or on homomorphic encryption and secret sharing [13], [19], [30], [36], [41]. The latter is more computationally efficient than the former, but incurs high communication costs since each user has to transmit their secret share with several other users. Such generic MPC protocols can be used for PDA, but since they are not optimized solely for aggregation, they are not as efficient as bespoke PDA solutions, such as Masquerade. Likewise, generic MPC schemes allow the participants to provide any input to the aggregation, whereas Masquerade protects against malicious inputs using the two non-interactive ZKP protocols discussed in section IV-D. Both MPC and the works in [20], [38] that use thresholding-based cryptography (such as Shamir’s secret-sharing [61]) remain susceptible to participant dropouts. Conversely, Masquerade is immune by design to participants intentionally leaving the computation. Finally, a critical benefit of Masquerade compared to generic MPC solutions is that our method offers public verifiability, where third parties can audit the committed result.

Shi et al. [63] proposed a PDA methodology based on random shares and distributed noise generation. The authors of [15] and [34] extended Shi’s protocol by handling client dropouts and enabling larger plaintext spaces, respectively. Likewise, the work in [20] is also based on secret-sharing techniques for privacy-preserving smart meter readings, yet it cannot tolerate participant dropouts. Leontiadis et al. [37] extended [34] and eliminated some of the trust assumptions that the latter requires. PUDA [38] is a pairing-based scheme that uses a trusted dealer to set up secret keys and enables public verification using homomorphic tags. While these aforementioned protocols support only quantitative variables and any client can corrupt the results, Masquerade offers robust protection against malicious clients, as well as support for both categorical and quantitative studies. The work in [10] features an efficient t-out-of-n secret sharing protocol, however, like most related works, it cannot tolerate malformed inputs from malicious participants. Trinocchio [60] outsources a computation from a single client to multiple workers in a privacy-preserving way and generates verifiable guarantees of the correct computation. Trinocchio additionally describes a multi-client version but this variant requires a special key generation and also is susceptible to malicious inputs: if any client provides incorrect information, the computation is aborted. Moreover, among the aforementioned works, only PUDA and Masquerade offer public verifiability.

Prio [16] is based on multiple non-colluding servers and is the most closely related to our scheme. It introduces a technique called secret-shared non-interactive proofs (SNIPs) to validate that clients’ inputs are the encryption of either a “0” or a “1”. To aggregate longer values than just one bit, Prio encodes the participants’ private data as a sequence of bits, where each bit is verified by a different SNIP, and computes
the sum of these encodings. Therefore, since SNIPs inherently prove one bit, to prove messages of bigger sizes the proof size increases linearly to the number of bits. Notably, the range proofs in Masquerade have constant size and time, whereas our set-membership proofs scale linearly to the set elements. For very small client inputs, the design of Prio enables fast participant timings: For instance, the authors report that for one-bit up to four-bit integers, the participant timing varies from 0.01 to 1 second, depending on the number of the multiplication gates on the SNIP circuit. We remark that Masquerade offers better scalability, as our plaintext dynamic range is in the order of thousands of bits (e.g., 2048 bits in out experiments), which can encode significantly more information. While the authors of Prio do not report any timings for categorical studies (which can be theoretically supported with modified proofs), their approach introduces additional overhead since each participant must prove that the sum of all the bits is exactly one. Finally, Prio does not operate on a public ledger and does not offer any public verifiability protections, contrary to our Masquerade protocol.

The work in [49] introduced zkLedger, a method that protects participants’ privacy and provides public verifiability. The participants in zkLedger rely on Pedersen commitments [54] and Schnorr-type ZKPs [59] to publish their transactions on a public ledger. Then an auditor can combine all the public commitments and verify their correctness. Likewise, in Masquerade we allow the analyst to audit the commitments, but since everything is public, any participant or third party can also audit the committed result. Nevertheless, zkLedger incurs significant overheads since transaction verification costs are quadratic to the number of participants, so practicality is significantly impacted in studies with more than 10-20 participants. Conversely, our experiments show that Masquerade scales gracefully for thousands of participants.

Table III summarizes all notable related works and compares with ours. Interestingly, these approaches use a broad range of cryptographic techniques, yet all of them have homomorphic properties. We observe that most of these constructions are tailored to quantitative studies, whereas only our protocol and Prio [16] provide encodings to represent categorical variables (e.g., Likert scales). Moreover, only PUDA, zkLedger, and Masquerade allow practical auditing and public verifiability, which is an important property for real-world applications. Although several works have focused on malicious participants intentionally dropping out of the protocol, only our protocol, SEPIA, Prio, and zkLedger protect against malicious inputs using non-interactive ZKPs; however, only Masquerade offers constant ZKP costs for quantitative studies.

The security and threat models of the aforementioned existing works vary based on the number of computing parties, the types of studies they support, and their robustness against malicious participants. The protocols in [15], [34], [63] use a single untrusted aggregator, and assume the aggregator obliviousness model which ensures that the aggregator learns only the sum of users’ inputs and nothing else. The authors of [38] extended the aggregator obliviousness model and introduce the concept of aggregate unforgeability, which assumes that one party performs the aggregation and another party decrypts and publishes the final result. Aggregate unforgeability ensures that the aggregator cannot convince the data analyzer to accept a bogus sum. Similarly to our work, in this model the two parties should never collude. In [10], the authors introduce three different security models, one which ensures security only against the clients, one which ensures security only against the server, and a mixed security model for a threshold of colluding clients with the server. The work in [13] uses the honest-but-curious model and is secure as long as no more than half of the computing nodes collude, while the authors of [30] use the Sharemind MPC framework with three honest-but-curious non-colluding nodes. Lastly, the protocol in [49] is secure against malicious input providers (e.g., banks) and keeps the transactions private as long as that the sender, the receiver, and the auditor will not collude, whereas the authors of [16] use a small set of servers (e.g., three) and protects the client privacy if not all the servers collude. It is evident that, like Masquerade, most related works rely on two or more non-colluding servers. However, as shown in Table III, only Masquerade offers public verifiability while protecting against malicious inputs and scales with an increasing number of participants.

VII. CONCLUDING REMARKS & FUTURE WORK

We present Masquerade, a new scheme for private data aggregation that ensures that participants’ inputs are kept private and the analyst only learns the final aggregate result. We have expanded our protocol to work both with quantitative and categorical variables, enabling a variety of different studies. Our two non-interactive ZKP protocols complement our scheme to defend against malicious participants by verifying that the encrypted data are well-formed and that each participant has limited influence on the protocol. Masquerade publishes our novel multiplicative homomorphic commitments on a ledger to enable public verifiability, which is an essential property for real-world applications where participants need to assess the correctness of the encrypted sum and verify that their response is included in the final result. Moreover, our protocol’s public verifiability allows the analyst to prove that the announced decryption of the encrypted aggregate is correct, which allows any participant to audit the final results. Our experiments demonstrate that Masquerade can scale gracefully to thousands of participants, and can further support smartphone clients.

Our future research will investigate the decentralization of the curator and the analyst to more parties. This will allow splitting the trust to more entities while still preserving the crucial property of public verifiability.

REFERENCES


Protocol 1 Zero-Knowledge Encryption-of-Zero Proof

Public Inputs: Let \( c \) be a Paillier ciphertext known to both \( \mathcal{P} \) and \( \mathcal{V} \). The soundness parameter \( t \) is the bit-size of the verifier challenge \( h \).

Private Inputs: \( \mathcal{P} \) knows the randomness \( \rho \) such that \( c = \rho N \mod N^2 \).

Goal: The prover \( \mathcal{P} \) convinces the verifier \( \mathcal{V} \) that \( c \) is the encryption of zero.

The protocol:

1) Proof:

a) \( \mathcal{P} \) generates a random \( \rho' \in \mathbb{Z}_N^* \) and computes \( a \leftarrow \rho' N \mod N^2 \) and sends it to \( \mathcal{V} \).

b) \( \mathcal{P} \) computes \( h \leftarrow \text{SHA256}(N, c, a) \).

c) \( \mathcal{P} \) computes \( z \leftarrow \rho' \ast \rho^h \mod N^2 \) and sends it to \( \mathcal{V} \).

2) Verification:

a) \( \mathcal{V} \) generates the randomness based on the transcript as \( h \leftarrow \text{SHA256}(N, c, a) \).

b) \( \mathcal{V} \) verifies that \( z = a \ast c^h \mod N^2 \).

APPENDIX A

ZERO-KNOWLEDGE PROOF PROTOCOLS

A. Zero-Knowledge \text{Enc}(0) Proof

Protocol 1 describes a non-interactive ZKP to prove that a given ciphertext is an encryption of zero. We adopt the protocol for proving ciphertext equality over \( N^{s+1} \) modulus with \( s = 1 \) from [18] and the Fiat-Shamir heuristic in the random oracle model to convert the ZKP to a non-interactive variant. Let \( c \) be a Paillier encryption of zero known both to \( \mathcal{P} \) and \( \mathcal{V} \) and \( \rho \) be the secret randomness used for \( c \) (i.e., \( c = \rho^N \mod N^2 \)). As described by the protocol, \( \mathcal{P} \) generates and sends \( a \) and \( z \) to \( \mathcal{V} \), who in turn can easily check that \( \text{Enc}_{pk}(0, z) = z \mod N^2 = a \ast c^h \mod N^2 \).

B. Zero-Knowledge Range Proof

Protocol 2 describes a non-interactive ZKP that a Paillier ciphertext \( c \) encrypts a message \( m \) from within a range of plaintexts \( R = \{0, \ldots, \ell \} \). We adopt the range proof from [40] and make it non-interactive using the Fiat-Shamir heuristic in the random oracle model to eliminate the interaction between the prover and the verifier. More specifically, in step 2), \( \mathcal{P} \) generates the randomness \( h \) based on the transcript of the protocol until that point (i.e., \( c_1^h, \ldots, c_{\ell}^h \)). In step 4), \( \mathcal{V} \) generates \( h \) herself based on the public transcript and continues with the verification of the proof. The soundness parameter \( t \) corresponds to the number of queries that the prover has to answer in order to convince the verifier, i.e., the probability that it answers all is at most \( 2^{-t} \).

C. Zero-Knowledge Set Membership Proof

Finally, in Protocol 3, we describe a non-interactive variant of a ZKP that asserts that a Paillier ciphertext \( c \) encrypts a message \( m \) from a public set of plaintexts \( S = \{m_1, \ldots, m_k\} \) [3]. Our proof uses the Fiat-Shamir heuristic in the random oracle model to eliminate the interaction between the prover and the verifier. In step 2), the prover generates the (unpredictable) challenge \( h \) as \( \text{SHA256}(u_1, \ldots, u_k) \) so that anyone can re-generate \( h \) from the \( u \) vector. If the protocol is iterated \( t \) times and \( m' \not\in S \), then the probability that a malicious
Protocol 2 Zero-Knowledge Range Proof

Public Inputs: Let \( \mathcal{R} = \{0, \ldots, \ell\} \) be a range of \( \ell + 1 \) messages and \( c = g^{m/N} \mod N^2 \) be the Paillier encryption of a secret message \( m \in \{0, \ldots, \ell\} \). Let \( t \) be the soundness parameter.

Private Inputs: \( \mathcal{P} \) knows secret message \( m \in \mathcal{R} \).

Goal: The prover \( \mathcal{P} \) convinces the verifier \( \mathcal{V} \) that \( c \) is the encryption of a message in \( \mathcal{R} \).

The protocol:

1) Setup:
   a) For \( i \in \{1, \ldots, t\} \), \( \mathcal{P} \) chooses random \( w_i^{1}, \ldots, w_i^{\ell} \leftarrow \{\ell, \ldots, 2\ell\} \) and computes \( w_i^{\ell} = w_i^{1} - 1 \).
   b) \( \mathcal{P} \) switches the values of \( w_i^{1} \) and \( w_i^{\ell} \) with probability \( 1/2 \) independently for each \( i \in \{1, \ldots, t\} \).
   c) For \( i \in \{1, \ldots, t\} \), \( \mathcal{P} \) computes \( c_i^1 = \text{Enc}_{pk}(w_i^{1}, \rho_i^{1}) \) and \( c_i^2 = \text{Enc}_{pk}(w_i^{\ell}, \rho_i^{\ell}) \), where \( \rho_i^{1}, \rho_i^{\ell} \leftarrow \mathbb{Z}_N \) are the randomness used in the Paillier encryption.

2) Commit: \( \mathcal{P} \) generates \( h \leftarrow \{0,1\}^t \) in the random oracle model as \( h \leftarrow H(c_1^{1}, c_2^{1}, \ldots, c_t^{1}, c_2^{2}) \).

3) Proof: For each \( i \in \{1, \ldots, t\} \):
   a) Depending on the value of \( h_i \), \( \mathcal{P} \) sets \( z_i \) as follows:
      \[
      z_i = \begin{cases} \left(w_i^{1}, \rho_i^{1}, w_i^{\ell}, \rho_i^{\ell}\right), & h_i = 0 \\ \left(j, m + w_j^{1}, \rho \cdot h_j^{1} \mod N\right), & j \in \{1,2\} \quad \text{and} \quad m + w_j^{1} \in \{\ell, \ldots, 2\ell\}, \quad h_i = 1. \end{cases}
      \]
   b) Finally, \( \mathcal{P} \) sends \( c_1^{1}, c_2^{1}, \ldots, c_t^{1}, c_2^{2} \) and \( z_1, \ldots, z_t \) to \( \mathcal{V} \).

4) Verification: \( \mathcal{V} \) generates \( h \) as in step 2 and then parses \( z_i \) according to the value of \( h_i \). For every \( i \in \{1, \ldots, t\} \):
   a) If \( h_i = 0 \), \( \mathcal{V} \) checks that \( c_i^1 = \text{Enc}_{pk}(w_i^{1}, \rho_i^{1}) \) and \( c_i^2 = \text{Enc}_{pk}(w_i^{\ell}, \rho_i^{\ell}) \) and that one of \( w_i^{1}, w_i^{\ell} \in \{\ell, \ldots, 2\ell\} \) while the other is in \( \{0, \ldots, \ell\} \), where \( z_i = (w_i^{1}, \rho_i^{1}, w_i^{\ell}, \rho_i^{\ell}) \).
   b) If \( h_i = 1 \), \( \mathcal{V} \) checks that \( \mathcal{V} \) computes \( h_i \) and \( v_i \) such that \( v_i = \rho \cdot h_i \cdot \rho \cdot h_i \mod N \) and sends \( v_1, \ldots, v_t, u_1, \ldots, u_k, h_1, \ldots, h_k \) to \( \mathcal{V} \).
   c) \( \mathcal{V} \) outputs 1 if and only if all of the above checks pass.

Prover convinces an honest verifier that \( m' \in S \) is \( 1/\mathcal{A}' \), where \( \mathcal{A} \) is the size of the challenge.

APPENDIX B

MASQUERADE SECURITY PROOF

Definition 1. Masquerade \( \mathcal{M} \) is a private data aggregation protocol which comprises four algorithms \( \mathcal{M} = (\text{KG}, \text{EPC}, \text{AG}, \text{DAP}) \), described in Fig. 2.

1) \( \text{KG} \) is a probabilistic, key generation algorithm, that takes the security level as input and outputs a pair \( (sk, pk) \), where \( sk \) is called the secret key and \( pk \) is called the public key. \( \text{KG} \) additionally outputs a prime number \( e \) and a public element of maximum order \( g_m \) as described in section IV-A.

2) \( \text{EPC} \) is a non-interactive protocol algorithm for encrypting, proving, and committing to participant data.

3) \( \text{AG} \) is a non-interactive protocol algorithm that takes as input the encryptions of the participants’ data \( \text{Pai}_m \) and the randomness \( r_i \) used for \( c_i \), and a zero-knowledge proof \( z_i \) that \( m_i \) is valid. \( \text{AG} \) verifies every \( z_i \), checks the correctness of each \( c_i \), and outputs accept or reject for each \( z_i \) and \( c_i \). Then, \( \text{AG} \) aggregates all the encryptions \( \text{Pai}_m \) whose \( z_i \) and \( c_i \) were valid. Finally, \( \text{AG} \) outputs the sum of the encryptions (i.e., \( \text{Pai}_m \)) and the sum of the corresponding randomness \( r_m \).

4) \( \text{DAP} \) is a non-interactive protocol algorithm for decrypting, auditing, and proving the final sum. First, \( \text{DAP} \) verifies that the sum of the public commitments \( c_i \) opens to the commitment of the sum (e.g., by committing using the provided \( r_m \) from \( \text{AG} \)) and outputs accept or reject. Next, only if the commitment was valid, \( \text{DAP} \) decrypts \( \text{Pai}_m \) and outputs the result \( s_m \). Finally, \( \text{DAP} \) outputs a non-interactive ZKP for the correct
decryption.
We require that for a given AG that follows the protocol and for all possible outputs of KG, AG outputs accept for honest participants and reject for malicious ones with overwhelming probability.

To analyze the security of this protocol we formally define the notion of security against direct attacks, and then explain why this protocol satisfies this definition.

**Attack Game 1** (Confidentiality). For Masquerade $M = (KG, EPC, AG, DAP)$ and a given adversary $A$ which can be either of the participants or an external party (but not the curator or the analyst since they are semi-honest as described in section III-B), the attack game against the participants’ data confidentiality runs as follows:

$A$ submits two distinct chosen plaintexts $m_0, m_1$ to the challenger. The challenger selects a bit $b \in \{0, 1\}$ uniformly at random and sends $\text{Pai}_m = \text{Enc}_{pk}(m_b, \rho), z_b = \mathcal{P}(m_b), c_b = \mathcal{C}(\text{Pai}_{m_b}, r)$ to $A$. $A$ is can to perform any number of additional encryptions, proofs, and commitments. Finally, $A$ outputs a guess for the value of $b$.

We say that $A$ wins the game if they have non-negligible advantage over random guessing, denoted by $M1_{\text{adv}}[A, M]$. $A$ wins the game if they can guess $b$ with probability $0.5 + \epsilon(N)$, where $\epsilon(N)$ is a non-negligible function in the security parameter $N$.

**Definition 2.** We say that Masquerade $M$ is secure against confidentiality attacks if for all efficient adversaries $A$, the $M1_{\text{adv}}[A, M]$ is negligible.

**Theorem 2.** Suppose that the Paillier cryptosystem is semantically secure against chosen-plaintext attacks (IND-CPA). Suppose that hash function $H$ used for the Fiat-Shamir heuristic is one-way and that the range proof and set-membership protocols are zero-knowledge. Finally, suppose that the commitment scheme (a) is hiding and (b) even when we partially open it by revealing the randomness, it still preserved the privacy of $m_i$ as the commitment is made to $\text{Pai}_{m_i}$. Then, the Masquerade protocol $M$ is secure against attack game 1.

**Proof.** To attack $M$, $A$ must be able to either break IND-CPA for the Paillier encryption or break the zero-knowledge property of the range-proof or the set-membership proof, both of which are impossible. Finally, as the commitment is made on a Paillier ciphertext which is probabilistic, even revealing the randomness, the remaining $(\text{Pai}_{m_i})^c \mod N^2$ is secure since it is an RSA encryption of a Paillier ciphertext (assuming that the factoring problem is hard if $N$ has 2048 bits or more).

**Attack Game 2** (Integrity). For Masquerade $M = (KG, EPC, AG, DAP)$ and a given adversary $A$, the adversary $A$ wins the game if at the end of the protocol the result is not the sum of the participants’ inputs. Below, we break down the attack game:

### 2.1 Numerical data:
An adversary $A$ that plays the role of a participant can affect the final sum if they can produce a plaintext $m'$ that is outside the range of valid plaintexts (i.e., $m' \not\in \{0, 1, \ldots, \ell\}$) and its corresponding ZKP $z = \mathcal{P}(m')$ so that the curator accepts $z$. We denote the probability that $A$ convinces the curator as $M2_{\text{adv}}[A, M] = \text{Prob}[\text{Verify}(\mathcal{P}(\text{Pai}_{m'}, m')) = 1]$.

### 2.2 Categorical data:
An adversary $A$ that plays the role of a participant can affect the final sum if they can produce a plaintext $m'$ that is outside the set of valid plaintexts (i.e., $m' \not\in \{m_1, m_2, \ldots, m_k\}$) and its corresponding ZKP $z = \mathcal{P}(m')$ so that the curator accepts $z$. We denote the probability that $A$ convinces the curator as $M3_{\text{adv}}[A, M] = \text{Prob}[\text{Verify}(\mathcal{P}(\text{Pai}_{m'}, m')) = 1]$.

### 2.3 Commitment:
An adversary $A$ that plays the role of a participant can affect the result of the protocol if they can trick an honest curator to accept an invalid commitment. This will result in the analyst aborting the protocol as the commitment to the final sum will not be equal to the product of the partial commitments published to the ledger by the participants. More formally, $A$ succeeds if they can find $m_1, m_2, r_1, r_2$ with $m_1 \neq m_2$ such that $\mathcal{C}(e, g_m, \text{Pai}_{m_1}, r_1) = \mathcal{C}(e, g_m, \text{Pai}_{m_2}, r_2)$ or find $m_1, m_2$ with $m_1 \neq m_2$ such that $c_1 = \mathcal{C}(e, g_m, \text{Pai}_{m_1}, r)$ and $c_2 = \mathcal{C}(e, g_m, \text{Pai}_{m_2}, r)$ and can decide with probability better than $1/2$ if $m_1$ is $c$ or $c'$. We denote $A$'s advantage as $M4_{\text{adv}}[A, M]$.

### 2.4 Decryption:
An adversary $A$ that plays the role of the analyst can manipulate the result of the protocol if they can deceive an honest auditor to accept a result that is not the correct decryption of $\text{Pai}_{\text{sum}}$. More formally, $A$ succeeds if they can create a ZKP $z$ so that $\text{Pai}_0$ is an encryption of zero. Recall from Section IV-C that $\text{Pai}_0 = \text{Pai}_{\text{sum}} \ast (\text{Pai}'_{\text{sum}})^{-1}$ and that since $\text{Pai}_{\text{sum}}$ and $\text{Pai}'_{\text{sum}}$ are public, the aggregator cannot pick an arbitrary value for $\text{Pai}_0$. Since $A$ published an incorrect decryption of $\text{Pai}_{\text{sum}}, \text{Pai}_0$ will not be an encryption of zero. We denote $A$'s advantage as $M5_{\text{adv}}[A, M] = \text{Prob}[\text{Verify}(\mathcal{P}(\text{Pai}_0, \text{Enc}_{pk}(0, \rho'))) = 1]$.

**Definition 3.** We say that Masquerade $M$ is secure against integrity attacks if for all efficient adversaries $A$, $M2_{\text{adv}}[A, M], M3_{\text{adv}}[A, M], M4_{\text{adv}}[A, M]$, and $M5_{\text{adv}}[A, M]$ are negligible.

**Theorem 3.** Suppose that both the non-interactive range proof (presented in protocol 2) and the set-membership proof (presented in protocol 3) are sound. Additionally, suppose that the commitment scheme is hiding. Then, the Masquerade protocol $M$ is secure against attack game 2.

**Proof.** To win the attack game 2 against the integrity of $M$, $A$ must be able to either find a collision on our commitment or break the soundness of the ZKPs (i.e., range proof in numerical studies or set-membership proof in categorical studies).

### 2.1 Numerical data:
Let’s assume that $m' \not\in R$, where $R = \{0, \ldots, \ell\}$. To break the soundness of the protocol
and prove that \( m' \in R \), a cheating prover \( \mathcal{P}' \) needs to provide accepting answers for both \( h_0 \) and \( h_1 \) for the \( i \)th ciphertext. This suffices since it implies that \( \mathcal{P}' \) can answer at most one of the \( h_i \) queries for each \( i \), and thus the probability that it answers all is at most \( 2^{-t} \).

### 2.2 Categorical data:

Let’s assume that \( m' \notin S \), where \( S = \{ m_1, \ldots, m_k \} \) and that a cheating prover \( \mathcal{P}' \) successfully completes an iteration of the protocol. For each \( j \in \{ 1, \ldots, k \} \), \( v_j = u_j (c/g^{m_j})^{h_j} \mod N^2 \).

Taking the partial logarithms, it follows that
\[
\log(1) = \log(u_j (c/g^{m_j})^{h_j}) \mod N \Rightarrow 0 = \log(u_j) + \log(g^{m_j})h_j \mod (N) \Rightarrow 0 = \log(u_j) + (m - m_j)h_j \mod N \text{ and since } m \neq m_j \mod N,
\]
\[
e_j = \log(u_j)/(m_j - m) \mod N \text{ for each } j \in \{ 0, \ldots, k \}.
\]

Finally, since \( h = \sum_j h_j \mod N \), it follows that \( h = \sum_j \log(u_j)/(m_j - m) \mod N \), which holds with probability at most \( 1/(2^t) \). Iterating the protocol \( t \) times, we have \( M3adv[A, M] = 1/(2^t) \), which is negligible.

### 2.3 Commitment:

We have proved in section IV-A the hiding property of our commitment scheme.

### 2.4 Decryption:

Assume that \( Pai_0 \) is not an encryption of zero. Then, a cheating prover \( \mathcal{P}' \) can successfully create \( z_0 \) with probability \( 2^{-t} \), which is negligible for sufficiently large \( t \).

### Appendix C

#### Real-World Applications

In Table IV, we present various real-world applications of Masquerade and some example instantiations for the curator, the analyst, and the participants. For instance, the course evaluation study can be either numerical (i.e., computing an average evaluation score for a class) or categorical (i.e., by using a Likert scale as mentioned in section III). In either case, the protocol can be executed with the University as the analyst, the Department of Education as the curator, and the students as participants. The rest of the applications can run in a similar manner.

We now focus on an auction application as it has an interesting use of our categorical encoding and it also demonstrates the need for public verifiability since all the participants want to verify that the auction was carried out correctly and that their bids were also counted. In a Vickrey auction (also known as sealed-bid, second-price auction [55]), bidders submit their private bids without knowing the bids of others. As in a traditional auction, the highest bidder wins, however, in a Vickrey auction the price paid equals the second-highest bid. Using our encoding for categorical data our protocol can support a Vickrey auction with \( K \) different bid options. After all the participants have submitted their private bids, the analyst decrypts the encrypted sum and determines the winning bid (i.e., the left-most category has highest value). Since the bids are private, the analyst uses Protocol 1 to prove the decryption of the winning bid (i.e., applying Alg. 4 in Fig. 3 to the winning ciphertext), and the winner can open her commitment in order to prove their bid and claim the prize. Finally, the amount paid by the winner is the second highest bid (public information to all since the sum is decrypted). In sealed-bid auctions, public verifiability is crucial so that any participant can audit the protocol and verify the results.

---

<table>
<thead>
<tr>
<th>Application</th>
<th>Application Type</th>
<th>Participants</th>
<th>Curator</th>
<th>Analyst</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course-evaluations</td>
<td>Numerical/Categorical</td>
<td>Students</td>
<td>US Department of Education, State, etc.</td>
<td>University</td>
</tr>
<tr>
<td>Smart-metering</td>
<td>Numerical</td>
<td>Households</td>
<td>AWS</td>
<td>Utility company</td>
</tr>
<tr>
<td>Auctions</td>
<td>Categorical</td>
<td>Participants</td>
<td>AWS</td>
<td>Auctioneer</td>
</tr>
<tr>
<td>Healthcare</td>
<td>Numerical/Categorical</td>
<td>Patients</td>
<td>Insurance company, federal agency, etc.</td>
<td>Hospital</td>
</tr>
<tr>
<td>Banking</td>
<td>Numerical/Categorical</td>
<td>Customers</td>
<td>Federal/state department</td>
<td>Bank</td>
</tr>
</tbody>
</table>