Privacy-Preserving Transactions With Verifiable Local Differential Privacy

Danielle Movsowitz Davidow*
Tel-Aviv University, Israel
dm4@mail.tau.ac.il

Yacov Manevich
IBM Research – Zurich, Switzerland
yacov.manevich@ibm.com

Eran Toch
Tel-Aviv University, Israel
erant@tauex.tau.ac.il

August 14, 2023

Abstract

Privacy-preserving transaction systems on blockchain networks like Monero or Zcash provide complete transaction anonymity through cryptographic commitments or encryption. While this secures privacy, it inhibits the collection of statistical data, which current financial markets heavily rely on for economic and sociological research conducted by central banks, statistics bureaus, and research companies. Differential privacy techniques have been proposed to preserve individuals' privacy while still making aggregate analysis possible. We show that differential privacy and privacy-preserving transactions can coexist. We propose a modular scheme incorporating verifiable local differential privacy techniques into a privacy-preserving transaction system. We devise a novel technique that, on the one hand, ensures unbiased randomness and integrity when computing the differential privacy noise by the user and on the other hand, does not degrade the user's privacy guarantees.

1 Introduction

1.1 Motivation

The issue of privacy holds significant importance and poses ongoing challenges in blockchain systems. This concern has become a substantial barrier for traditional financial institutions seeking to adopt blockchain technologies [26]. Initially, blockchain protocols offered only limited pseudo-anonymity. However, user concerns have given rise to privacy-preserving blockchain systems such as

*The work was done during an internship at IBM Research – Haifa.

Monero [42], ZCash [5], and Twilight [20]. These systems aim to provide increase users' anonymity and conceal transaction details. For instance, ZCash [5] allows users to encrypt transaction data, safeguarding the sender's, recipient's, and transaction amount's privacy. Additionally, ZCash ensures the validity of transactions without revealing any supplementary information beyond the transaction's legitimacy. These advancements within blockchain systems show a proactive approach to addressing privacy concerns, facilitating broader adoption of blockchain technology.

However, many pivotal applications such as statistics gathering [29], federated model learning [32, 31], and anomaly analysis [30] necessitate the acquisition of aggregated models from the shared data of numerous users while still maintaining user confidentiality. While such aggregate and statistical models can yield benefits to the market as a whole [43], it is equally crucial to ensure that data collection is conducted to minimize privacy threats to the users. In particular, we aim to inhibit the ability of the data analyst to link multiple data points of the user [19] or retrace their real identity via third-party datasets [17]. If these risks aren't adequately addressed, users may opt to evade participation in data collection initiatives or provide misleading information that could lead to biased models.

Central Bank Digital Currencies (CBDCs) present a significant use case for the aggregation of statistical models. Central banks across the globe have exhibited a growing interest in developing and issuing CBDCs, which represent a digitized form of a nation's fiat currency and are intended to be widely available to the general public. Blockchain-based CBDCs are designed to align with the requirements of regulated financial institutions. Consequently, these institutions favor *permissioned* blockchain platforms that mandate identity management, audibility, and non-deniability to comply with government-established monetary regulations.

However, the transactions resulting from existing privacy-preserving systems tailored for financial institutions [3] appear random, preventing extraction of meaningful insights through observation. They prioritize the properties of *unlinkability* and *untraceability* to protect user privacy. Unlinkability ensures that user actions cannot be easily linked together, while untraceability maintains the anonymity of each transaction's sender and recipient. These systems safeguard sensitive user information and transaction history by upholding these properties. Therefore, central banks cannot derive valuable insights and may struggle to meet regulatory requirements.

Local differential privacy [33] potentially solves this by adding random "noise" to the data. This noise ensures that no specific user can be identified from the processed data, a privacy guarantee laid out by Dwork et al. (2006) [23]. Thus, financial institutions can gather and analyze aggregated data without risking the exposure of sensitive information tied to individual transactions. However, a challenge arises here: while local differential privacy empowers users to introduce noise to their data, some may distort this feature by injecting biased noise into it, which may adversely affect the integrity of the information collected [12, 15]. Therefore, in blockchain environments, it's crucial to have a mechanism verifying the correctness of introduced noise when implementing local differential privacy.

1.2 Verifiable Differentially Private Transactions

In this paper, we introduce the Verifiable Differentially Private (VDP) transaction scheme, designed to ensure user privacy and data integrity during the data collection process for aggregated models. Our innovative scheme bolsters the capabilities of any privacy-preserving transaction system that maintains unlinkability and untraceability, such as Zcash [5]. With this enhanced setup, a third-party entity, like a data analyst, can gather aggregated data while simultaneously preserving user privacy and supplying plausible deniability.

Our approach stands out from other privacy-preserving transaction methodologies by incorporating *verifiable* Local Differential Privacy (LDP) utilizing zero-knowledge proofs. This integration forms an integral part of the privacy-preserving transaction system. By adopting LDP methods [33], data analysts, such as the statistical bureau of a central bank collecting financial information, can provide users with plausible deniability for their data that can be represented as binary variables. These variables can encompass a wide range of characteristics, such as non-profit organization affiliation or location in the U.S. Furthermore, including zero-knowledge proofs allows data analysts to verify the accuracy and integrity of the disclosed data.

Local Differential Privacy (LDP) approaches typically require users to generate random numbers that are then used to create noise in the data [33]. Keeping these random numbers confidential is crucial for maintaining user privacy. At the same time, it's also important that the process of creating random numbers and generating noise from them is unbiased. This means that even if someone involved in the process - either the user or the data analyst - has malicious intent, the LDP approach should still meet the following requirements: (i) The generation of noise cannot be influenced by either the user or the data analyst; (ii) The user can provide proof, without compromising privacy, that the noise is not biased; (iii) Once the random numbers for a particular input are calculated, the user cannot change these numbers or add a different level of noise. To meet these requirements, we introduce the VerRR algorithm, which is a straightforward verifiable LDP mechanism based on the concept of 'randomized response' [24].

1.3 Related Work

In the following sections, we outline the different approaches for verifiable differential privacy employed by various works and highlight the differences from our technique.

1.3.1 Central Differential Privacy

When applying Differential Privacy (DP) techniques for preserving privacy, traditional methods often rely on the assumption that the process of noise generation and application is intrinsically reliable. However, recent scholarly work [13, 14] has underscored the naivety of this assumption. These studies expose DP's vulnerability to a range of manipulation attacks, thereby threatening the integrity of the data under examination.

Several studies have proposed strategies to counter various manipulation attacks. The key aim across all these works is to generate and apply the noise

in a way that can be verified. Within the centralized DP model, studies such as [7, 38, 41] have zeroed in on a series of use cases. In these instances, the data owners forward their data—either in plain text, encrypted or in a secret-shared format—to a single party or a consortium of parties who together form an entity termed a "curator". This curator subsequently applies noise and generates an anonymized, privacy-preserving data set suitable for further analysis.

Secure Multi-Party Computation (sMPC) In the context of secure multi-party computation (sMPC), a user secretly shares their input and distributes the shares across several servers. As proposed in [7], if at least one of the servers collecting the data is honest, the resulting computation is either correct or detected to be incorrect. Unfortunately, such a single-client-multi-server model does not fit the use case of our work since we operate under the assumption that the data analyst, who also serves as the curator, could potentially have malicious intentions and is not divided into separate entities.

Zero-Knowledge Proofs In the study by Narayan et al., the curator creates a database comprising different users' data elements and subsequently publishes a cryptographic commitment to the entire database [38]. When a data analyst sends a query interested in specific statistical properties, the curator executes the differential privacy function. In addition, the curator generates[1] a Non-Interactive Zero-Knowledge (NIZK) proof, validating that the result aligns with the prior commitment, and sends both the result and the proof back to the data analyst. The data analyst then verifies the NIZK against the commitment to ensure that the differential privacy mechanism was correctly executed.

The most notable downside of the naive zero-knowledge approach employed by previous works is that it is assumed that the noise was sampled correctly and without bias. However, if the party that produces the NIZK is malicious, it may use far from random noise and thus either harm the privacy of the users or skew the results towards a value it favors. In contrast, in our technique, we ensure that some of the randomness is sampled by the user itself (unlike the work of [41] where the curator randomly samples the randomness) and some by the analyst, thus producing noise that is non-biased.

1.3.2 Local Differential Privacy Mechanisms

LDP mechanisms allow for developing a randomized response that is operated locally, making it fit for distributed blockchain environments. However, ensuring data integrity is a more challenging task since data manipulation occurs locally. Research, such as [2, 35], concentrates on making adjustments to existing LDP mechanisms to make them verifiable. This allows data analysts to verify how users introduce noise to the data.

The study by Kato et al. employs cryptographic randomized response techniques to validate existing LDP mechanisms. Their approach ensures the complete execution of privacy protection on the client side, without sacrificing user privacy [35]. However, their method's primary drawback is its interactive nature. Interactive methods are typically more time-consuming and resource-intensive

[1] For ease of explanation, we mention the curator as the producer of the NIZK, but in the cited study, this task is delegated to a separate party: the analyst.

than their non-interactive counterparts. They necessitate several communication exchanges and direct engagement with the data analyst. Furthermore, interactive methods do not support throttling. Hence, during periods of high network usage, while non-interactive systems can manage the demand by forming a queue that will eventually be cleared unbeknownst to the users, interactive systems may freeze, forcing users to endure an uncertain waiting time. Our proposed scheme, on the other hand, is non-interactive, overcoming these limitations.

Garrido et al. follow an approach based on zero-knowledge Succinct Non-interactive ARgument of Knowledge (zk-SNARKs) to adapt the implementation of select LDP mechanisms to a verifiable computation technique to prove the correctness of a differentially private query output [36]. However, their technique does not uphold the unlinkability and untraceability properties needed in a privacy-preserving transaction system. These properties are not upheld since the prover (i.e., the user) signs the randomness used as the base of the LDP mechanism with their private key, and the verifier (i.e., the data analyst) needs to know the prover's public key to verify the integrity of the response. By knowing the prover's public key, the data analyst can later connect the generated randomness and the transfer it was used in, thus revealing the identity of the user making the transfer.

Outline. The remainder of the paper is structured as follows. In Section 2, we introduce some preliminary concepts and provide the relevant background regarding blockchain-based privacy-preserving transaction systems and verifiable differential privacy. We present a high-level overview of our scheme in Section 3, and dive into the scheme's full details in Section 4. In Section 5, we evaluate the performance of our scheme. We discuss design choices in Section 6 and summarize our contribution in Section 7.

2 Preliminaries

2.1 Blockchain-Based Privacy-Preserving Transactions

Blockchain transaction systems are typically modeled using two approaches: (i) The account model, used in Ethereum [46]. This model is more user-friendly as it presents an abstraction of accounts and balances; (ii) The Unspent Transaction Output (UTXO) model, used in Bitcoin [37] and Zcash [5]. This model presents an abstraction of a plurality of coins, each with a balance and a condition for spending[2]. While it is considered an open problem how to "hide" a transaction's sender in an account model [28] [3], "hiding" the sender in the UTXO model is easy [5]. Thus, this work focuses on the UTXO model.

In a privacy-preserving transaction system using the UTXO model, when a sender transfers funds to a recipient, to preserve the privacy of the transaction, three things need to be hidden: (i) The sender's identity, (ii) The recipient's

[2]In most cases, presenting a signature that is verifiable under a public key that its hash is encoded in the coin

[3]In an account model, hiding the sender involves hiding it within a K sized anonymity set [11], while a UTXO based approach hides the sender within all possible senders in the system.

identity, and (iii) The transferred amount. The transferred amount is usually hidden through a cryptographic commitment and a zero-knowledge proof proving that the sum of input coins is greater or equal to the sum of the output coins. The recipient's anonymity is ensured by creating outputs that either have one-time public keys [42] or by hiding the output token's identity by having the coin itself be a commitment [5] to its properties (e.g., owner, amount). Maintaining the sender's anonymity varies widely across techniques; Monero [42] uses ring signatures to hide the source address among a randomly picked set of potential source addresses, while Zcash uses a Merkle tree and zero-knowledge proof of membership to obscure the spent token.

Our work is based on Zcash's approach to concealing sender, recipient, and amount but is flexible enough to incorporate other privacy-preserving transaction systems that encode the entire token as a commitment. This includes the model proposed by [4], which is better suited for CBDCs and permissioned settings, replacing the Merkle tree membership proof with proof of knowledge of a signature of an entity that certifies the coin. Zcash terminology is used throughout the paper.

2.2 Differential Privacy

Differential Privacy (DP) [23, 22] is a formal notion of privacy designed to allow learning helpful information about a population while learning as little as possible about an individual. DP guarantees that the presence or absence of any specific individual in a dataset does not affect the query output results of that database. To achieve this privacy requirement, DP models change the data by using some randomness that masks the user identity.

There are two main models of DP: the centralized model and the local model. In the centralized model, sensitive data is collected centrally in a single dataset, and a trusted data curator can access the raw data. Each user sends their data to the curator without noise in this model. Since we assume analysts requesting access to this dataset are malicious, the curator is responsible for correctly executing the differentially private mechanisms the analysts specify.

Definition 1 ($\epsilon-$differentially privacy [22]) *A randomized algorithm $\mathcal{K}$ is $\epsilon-$differentially private if for all data sets D_1 and D_2 differing on at most one element, and all $S \subseteq Range(\mathcal{K})$, $P[\mathcal{K}(D_1) \in S] \leq e^{\epsilon} P[\mathcal{K}(D_2) \in R]$.*
The probability is taken over the coin tosses of $\mathcal{K}$.

Local DP models can be based on a *randomized response* mechanism as was first proposed by Warner in 1965 [45] and formalized in the context of learning by Kasiviswanathan et al. [34]. In this model, the data curator and the analyst are often the same, and no trusted third party exists that holds the data and executes the mechanism. Therefore, the user makes data differentially private before sending it to the analyst, ensuring that even if an adversary gains access to the personal responses of individuals in the database, it will not be able to learn much about the individual's data.

Definition 2 (Local randomizer [24]) *An $\epsilon-$local randomized $R : \mathcal{X} \to W$ is an $\epsilon-$differentially private algorithm that takes a database of size $n = 1$ as input.*

We incorporate local DP techniques into our scheme since our work is based on a privacy-preserving transaction system in which we assume that any parties (i.e., the users or the data analyst) may be dishonest but not both. Therefore, an honest user cannot trust the analyst to properly blind their data without leaking it.

2.3 Randomized Response

Dwork and Roth presented in [24] a variant of the randomized response mechanism, in which a user answers a "Yes" or "No" question as follows:

1. Flip a coin.
2. If **tails**, then respond truthfully.
3. If **heads**, then flip a second coin and respond "Yes" if heads and "No" if tails.

In this algorithm, the user flips two coins before returning the response. Flipping two coins creates uncertainty about the true answer. This uncertainty is the source of privacy, as it gives plausible deniability to the data subject. In Appendix A, we prove the randomized response algorithm described above satisfies $\ln 3$-differentially privacy.

We note that when using the randomized response technique, the number of "Yesses" and "Noes" depends on the result of tossing the coin once or twice. Therefore, if an analyst wants to estimate the true number of "Yesses", it would need to analyze the randomness in the randomized response algorithm and estimate how many of the "Yesses" are truthful responses and how many are "fake", and are a result of the random coin flips.

2.4 Basic Cryptographic Building Blocks

The following section extensively explains zero-knowledge proofs, which are crucial in our scheme. We denote the symbol λ to represent the system-wide security parameter.

2.4.1 Commitment Schemes

A commitment scheme is a protocol between a sender and a receiver defined by three probabilistic polynomial time algorithms: $\langle Setup, Commit, Open \rangle$. The sender and receiver each invoke the *Setup* operation with the desired security parameter and then get back public parameters to be used for *Commit* and *Open*. The sender uses the *Commit* operation to encode a message m and get back a *commitment* to m. The sender then sends the commitment to the receiver. At a later stage, the sender may convince the receiver that the commitment encodes the message m by interacting with the receiver via the *Open* operation. In order for the commitment scheme to be useful, it needs to satisfy two security properties:

- **Hiding**: For any probabilistic polynomial time adversary $\mathcal{A}$ there exists a negligible[4] function ϵ such that for every two messages m_0, m_1 chosen

[4] A function $\epsilon : \mathbb{N} \to \mathbb{R}$ is negligible if for every positive polynomial $p(n)$ there exists an N such that $\forall n > N$: $\epsilon(n) < \frac{1}{p(n)}$

by $\mathcal{A}$ and a commitment $cm \leftarrow Commit(m_b)$ for $b \in \{0,1\}$ it holds that: $Pr[b \leftarrow \mathcal{A}(pp, cm)] - \frac{1}{2} \leq \epsilon(\lambda)$.

- **Binding**: For any probabilistic polynomial adversary $\mathcal{A}$ there exists a negligible function ϵ such that for every m and $cmt \leftarrow Commit(m)$ and every $m' \neq m$ chosen by
  $\mathcal{A}$: $Pr[Open(pp, cmt, m) = m'] \leq \epsilon(\lambda)$.

A commitment can either be information-theoretic binding or hiding, and it can be computationally binding or hiding (but cannot be both information-theoretic binding and hiding). In our scheme, the commitment does not need to be homomorphically additive or have any special property, and as such, it is completely pluggable.

2.4.2 Digital Signatures

A digital signature scheme is a triple $\langle Gen, Sign, Verify \rangle$ of probabilistic polynomial time algorithms. $Gen(1^\lambda)$ outputs (sk, pk) a private key and a public key respectively. For a message m, $Sign(m, sk)$ outputs a signature σ. For a message m, a public key pk and a signature σ, $Verify(pk, m, \sigma)$ accepts or rejects. The security property we require from a signature scheme is:

- **Unforgability**: For every probabilistic polynomial time adversary $\mathcal{A}$ with access to a signing oracle $\mathcal{O}$ which replies to queries denoted by a set Q there exists a negligible function ϵ such that $Pr[Verify(pk, m, \sigma) = 1 \wedge m \notin Q] \leq \epsilon(\lambda)$.

As in the case of the commitment scheme, the signature scheme employed by our protocol can vary, and its choice is insignificant.

2.4.3 Public Key Encryption

An encryption scheme is a triple $\langle Gen, Enc, Dec \rangle$ of probabilistic polynomial time algorithms.
$Gen(1^\lambda)$ outputs (sk, pk) a private key and a public key respectively. For a message m, $Enc(m, pk)$ outputs a ciphertext c. Given a ciphertext c and a private key sk, $Dec(c, sk)$ outputs a message m. For every public key pk with a corresponding sk and message m it holds that $Pr[Dec(Enc(m, pk), sk) = m] = 1$. The security property we require from an encryption scheme is:

- **Indistinguishablility**: For every probabilistic polynomial time adversary $\mathcal{A}$ and every pk generated by $Gen(1^\lambda)$ and for every two messages m, m' there exists a negligible function ϵ such that $Pr[\mathcal{A}(Enc(m, pk)) = 1] - Pr[\mathcal{A}(Enc(m', pk)) = 1] \leq \epsilon(\lambda)$.

Our protocol does not require any stronger property for the encryption scheme, such as resistance against chosen ciphertext attacks (CCA1 and CCA2).

2.5 Zero-Knowledge Proofs

For a language $\mathcal{L} \in NP$, denote the relation $R_\mathcal{L}$ to be the pairs (x, w) of statements and witnesses for x being in $\mathcal{L}$.

A zero-knowledge proof is a protocol between a prover P and a verifier V in which the prover can convince the verifier that it knows a witness w for a statement x if and only if $(x, w) \in R_{\mathcal{L}}$, and the verifier learns nothing from the protocol.

More formally, a pair of algorithms (P, V) is a zero-knowledge proof system for a language $\mathcal{L}$ if the following three conditions hold:

- **Completeness**: For every $x \in \mathcal{L}$ there exists w such that $Pr\left[\langle P(w), V\rangle(x) = 1\right] = 1$.
- **Soundness**: For every $x \notin \mathcal{L}$ and every P^* and every w there exists a negligible function ϵ such that $Pr\left[\langle P^*(w), V\rangle(x) = 1\right] \leq \epsilon(\lambda)$.
- **Zero-knowledge**: For every probabilistic polynomial time V^* there exists a probabilistic polynomial time simulator S such that for every $x \in \mathcal{L}$: $View_{V^*}^{P}(x) \equiv S(x)$.

While zero-knowledge proofs are not restricted to statements belonging to languages in NP, they are often used for such, especially for languages where verification of $(x, w) \in R_{\mathcal{L}}$ is efficient and can be modeled as a boolean or arithmetic circuit, but an efficient algorithm to find w for a random x is not known. A prominent example is: $(x, w) \in R_{\mathcal{L}} \Leftrightarrow f(w) = x$ where f is a cryptographic hash function.

In an anonymous set membership, given a set of items found as leaves in a Merkle tree, one can prove knowledge of an item in the Merkle tree by proving in zero-knowledge a path comprised of hash pre-images from one of the leaves to the root of the Merkle tree. Such a technique is the cornerstone behind the privacy-preserving cryptocurrency Zcash [5]. In Zcash, a sender wishing to hide the payment source simply creates a zero-knowledge proof that they use a coin that was added to a Merkle tree. The Merkle tree's leaves are all coins added as part of past transactions.

Zero-knowledge proof schemes come in different forms, each with its strengths and weaknesses. In this work, we focus on schemes ideal for privacy-preserving payments, and thus we require the scheme to be non-interactive, efficiently verifiable, and have a small proof size (succinct). A scheme with such properties is termed a zk-SNARK (Zero-Knowledge Succinct Non-interactive ARgument of Knowledge).

3 Overview of the VDP Transaction Scheme

Our privacy-preserving, verifiable, and differentially-private transaction scheme expands the functionality of any given privacy-preserving transaction system (e.g., Zcash [5]) by enabling a third party (e.g., a data analyst) to collect information about transaction attributes while preserving user privacy.

Before elaborating on the participants, components, and transaction flow, we reiterate the scheme's primary objective. The two main actors in our setting are a **user** with potentially sensitive information they are unwilling to disclose without plausible deniability and a **data analyst** who aims to create a statistical model by aggregating information gathered from user transactions. The users conduct transactions of a privacy-preserving nature. These transactions reveal nothing about the transactor's identity, the recipient's identity, or the properties

of the transaction (such as the amount transferred). Each transaction a user performs is accompanied by an additional Differentially Private Attribute (DPA) which results from applying an LDP algorithm to the transaction's private data[5]. The DPA can embed details like the sender's non-profit status, enabling the data analyst to tally transactions made by Blockchain users from both profit and non-profit sectors.

An LDP mechanism generates the differentially-private attribute that involves sampling randomness and using it for noise generation. The randomness must be kept secret from anyone but the user itself, lest a curious analyst can peel off the noise and be able to reconstruct the original value. At the same time, the scheme should force the user to generate randomness in a non-biased manner. Suppose the user is free to choose the randomness it uses. In that case, it can manipulate the result of the LDP mechanism making the data too "noisy", thus affecting the integrity of the data collected by the analyst.

Consequently, we seek a scheme with the following properties: (i) Neither the user nor the analyst can bias the generation of the noise used for the LDP mechanism; (ii) The user can prove in a privacy-preserving manner that the noise is non-biased; (iii) Once the randomness used to create the transaction's noise is computed, the user cannot compute new randomness or add a different amount of noise. We prove that our scheme fulfills all three aforementioned properties in Section 4.4.1. Another essential property of our scheme is preserving user privacy, meaning that once the transaction is complete, the analyst cannot identify the user who initiated the transaction. We prove this property in Section 4.4.2.

3.1 Participants

Our scheme involves the following participants:

Users. Users can exchange tokens within the system using transfer transactions, which are recorded on the blockchain. We presume that users possess specific individual attributes that they wish to disclose confidentially to the data analyst. These characteristics can cover a wide range of areas, for instance, signifying if the user is based in the U.S., is exempt from tax, or any other pertinent information that can be denoted as a binary variable.

Data analyst. This entity has the authority to collect aggregated information from the private attributes and activities of the users in the system. We assume that the analyst is only interested in analyzing data using statistical models regarding the system as a whole and is not interested in learning about specific individuals in the system.

Registration authority. This is a privileged entity in charge of registering all system users. For each user, it issues a long-term credential (i.e., a certificate) that binds the user's public key to its attributes. The same attributes will later be used as input to the LDP algorithm.

[5]In this work, we focus on the randomized response mechanism applied to one binary attribute. However, we argue that our ideas can be extended to randomized response vectors applied to multiple and non-binary attributes.

3.2 Threat Model

Our scheme is designed as a protocol that considers the interaction between two parties, the user and the data analyst, both of whom may exhibit dishonest behavior and possess conflicting objectives.

The first threat arises from the data analyst's desire to compute aggregated information derived from the user's transactions while potentially disregarding the user's privacy. Indeed, the analyst may deviate from the protocol in an attempt to link between the transaction and the user who submitted it [6].

The second threat arises from users who prioritize their privacy at the expense of providing misleading or corrupted data, which can undermine the integrity of the analysis conducted by the analyst. There are two primary ways in which users can launch such attacks: (i) Users can intentionally introduce randomness or noise manipulatively, rendering the data unreliable for generating unbiased and trustworthy aggregate statistical estimations [16]; (ii) Data poisoning attacks targeted at DP mechanisms. Data poisoning attacks involve adversarial manipulation of the input in order to influence the final aggregate result.

Since the analyst is interested in collecting statistics over attributes that correspond to the users themselves and for each transaction, such as whether the user is a non-profit or not or whether the user is based in the U.S., it is crucial that a user cannot mutate their attributes as they see fit. We guarantee this by requiring identities to be issued by the registration authority. Therefore, our scheme operates in the permissioned setting, whereas in a permissionless setting, each user is free to choose their own attributes.

Both the user and the analyst may deviate from the protocol. This introduces an element of uncertainty, as a party adhering to the protocol cannot definitively determine if the other party is also following protocol or diverging from it. Therefore, we aim to devise a protocol that protects the interest of any of the two parties as long as that party is honest. We note that such a protocol would fit the aforementioned threat model where both parties can deviate from the protocol, as each party can be assured that if it correctly follows the protocol, it will be protected from the misbehavior of the counter-party.

As we will see in the security analysis in Section 4.4, the focus on protecting the interests of honest parties lets us assume knowledge about the probability distribution of certain messages sent by honest parties, which then will set the probability distribution of messages sent by the counter-party regardless of whether it incorrectly samples its randomness.

In our protocol, the user initiates the protocol by sending a message to the data analyst, and the data analyst responds with its message. We consider a scenario where the data analyst does not respond to the user's message or sends back information that the user considers invalid as a scenario in which the data analyst aborts the protocol. Similarly to [4, 5], we consider network-level privacy out of scope and assume that the analyst cannot de-anonymize the user from inspecting the source of its network connection.

3.3 Components

Our scheme comprises three modular components: (i) A protocol to obtain and bind randomness. (ii) A verifiable LDP mechanism. (iii) An expanded privacy-

preserving transfer that includes verifiable differentially-private data.

Obtain and bind randomness protocol. This protocol is a privacy-preserving verifiable joint random number generation protocol between the user and the analyst. The protocol uses a serial number created by the user to bind a set of unspent input random seeds used in a specific transfer to their corresponding jointly generated randomness. The analyst uses this serial number to verify that only one randomness is created for each set of unspent inputs.

Verifiable LDP mechanism. This mechanism makes the user's attributes differentially private before they are disclosed to the analyst. For simplicity, in this work, we use the basic *randomized response* mechanism described in Section 2.3 to make a single binary attribute differentially private.

Expanded privacy-preserving transfer. Our verifiable differentially private (VDP) transfer expands the transfer transaction defined by the underlying privacy-preserving transaction system. As we explained in Section 2.1, our scheme works with any transfer mechanism that: (i) Encodes tokens as commitments to properties of the token (token value, owner, and random seed are all part of the input to a cryptographic commitment scheme). (ii) Uses serial number exposure as a double-spending prevention (for a random seed ρ, the serial number is $PRF\left(\rho\right)$).

To verify the correctness of the randomness used as the source of randomness in the LDP mechanism and to ensure that the analyst cannot link a specific transfer to its sender, our VDP transfer uses two serial numbers. The user creates the first serial number during the *obtain and bind randomness* protocol. The second serial number, also created by the user, is used to prove the correctness of the randomness used by the verifiable LDP mechanism. From the analyst's point of view, the second serial number will only be accepted if the first serial number was previously observed (without being able to link the two together).

The analyst can verify that it has previously seen the first serial number because both numbers have the same precursor, a set of unspent input random seeds. The generated serial numbers satisfy the following security properties: (i) They are collision resistant – two different sets of unspent tokens produce two different serial numbers. (ii) They are deterministic – the same set of unspent tokens will always produce the same serial number. (iii) They are unforgeable – only the user who owns the unspent tokens can produce a valid serial number. Although both serial numbers are computed on the same set of unspent inputs, the analyst cannot link them to each other thanks to their construction. The unspent input seeds are passed through Pseudo-Random functions with different keys and hence are computationally unlinkable.

Additionally, our transfer uses zk-SNARK proofs to verify the integrity of the data disclosed by the user (i.e., the DPA). The proofs prove that the disclosed DPA matches the user's original attributes and is created using the jointly generated randomness as the base of the randomness used in the LDP mechanism.

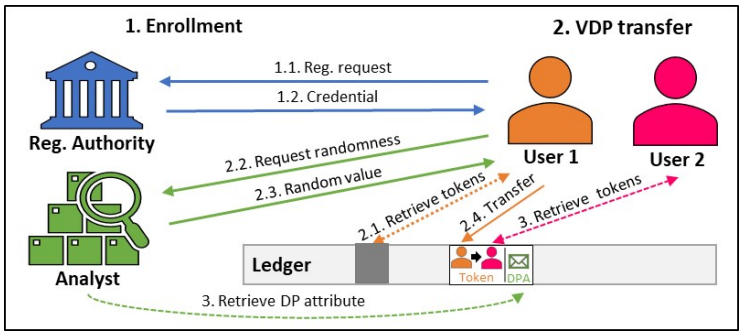


Figure 1: Overview of the VDP Transaction scheme.

3.4 VDP Transaction Flow

The VDP transaction scheme is illustrated in Figure 1. At first, the user contacts the registration authority (1.1) and uses a registration protocol to get a long-term credential (1.2). From then on, the user can use this credential as input for every VDP transfer transaction.

The VDP transfer is comprised of the following stages: The user retrieves its tokens from the ledger (2.1). The user contacts the data analyst (2.2) and executes the BindRandomness protocol. Thus, the user obtains a verifiable random value (2.3). The user executes the VDP transfer algorithm in three stages. First, the user adds noise to their attributes using the verifiable LDP mechanism (creates the DPA) and encrypts the result using the data analyst's public key. Then, the user computes the zk-SNARK proofs needed to verify the jointly generated randomness and the integrity of the DPA. Finally, the user uses the underlying transfer scheme in a black box manner to create a new unspent token. The resulting transfer transaction and the DPA encrypted with the analyst's public encryption key are posted on the ledger (2.4). Finally, the analyst (3) decrypts the encrypted DPA and includes it in its statistical computation, and the second user (3) (i.e., the recipient of the transaction) can now use the newly created token.

4 The VDP Transaction Scheme

This section details the complete VDP transaction scheme.

4.1 The BindRandomness Protocol

The BindRandomness protocol presented in Figure 2 is a privacy-preserving verifiable protocol executed between the user and the analyst. This protocol has two main goals: (i) Obtaining an unbiased random value jointly generated by the user and the analyst. (ii) Computing a verifiable and unlinkable serial number. This serial number will enable the user to later prove in zero knowledge that they know a random value jointly generated with the analyst to be used as a source of randomness for a randomized algorithm.

The BindRandomness protocol takes as input a security parameter λ and a vector $\vec{\rho} = (\rho_1, \ldots, \rho_m)$ s.t. $\forall i \in [m] \colon \rho_i \in \{0,1\}^\lambda$.

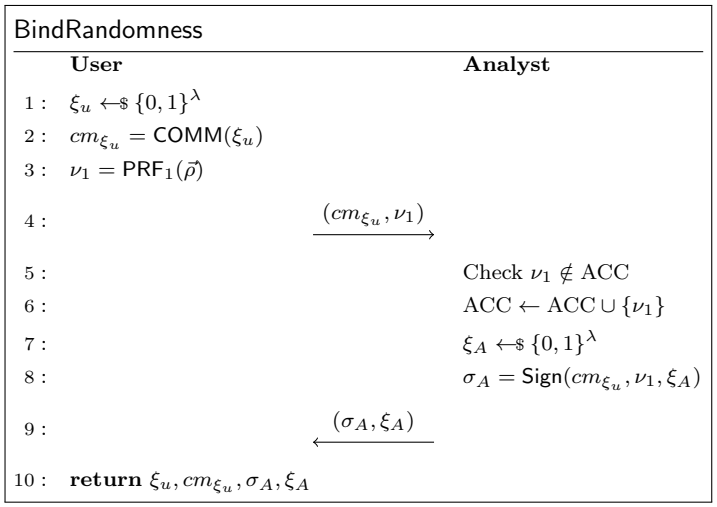


Figure 2: The BindRandomness protocol.

$\vec{\rho}$ represents m distinct seeds of unspent input tokens, and outputs two random values ξ_u, ξ_A, a commitment cm_{ξ_u}, and a signature σ_A.

In the initial stages of the protocol, the user samples a random value ξ_u, commits to it using $\mathsf{COMM}(\xi_u)$, and computes the serial number ν_1 with $\mathsf{PRF}_1(\vec{\rho})$. The user then sends the commitment cm_{ξ_u} and the serial number ν_1 to the analyst. The analyst checks if ν_1 was observed before and if so it aborts. Otherwise, the analyst will add ν_1 to some accumulator ACC, and continue on with sampling a random value ξ_A and signing $(cm_{\xi_u}, \nu_1, \xi_A)$ to obtain σ_A. The protocol ends with the analyst sending the user ξ_A and the signature σ_A. The user verifies the signature and discards ξ_A if the signature is found to be invalid.

After the protocol's execution, the analyst has a value ν_1 in its accumulator, which is correlated to m unspent tokens that the user may use in a future transaction. By signing the accompanied commitment of the user cm_{ξ_u} alongside ξ_A, the randomness picked by the user, as well as the randomness picked by the analyst are both indirectly bound to $\{\rho_i\}_{i=1}^m$ the seeds used for the unspent input tokens. As we will show in the next section, this is a crucial part of the security of our scheme: If the user used an unspent input's seed ρ_i to obtain randomness from the analyst, it must also use the corresponding unspent input in a transfer in order to use the randomness for the LDP computation.

4.2 The VerRR Mechanism

The VerRR algorithm is a simple, verifiable LDP mechanism based on *randomized response* presented in Section 2.3. This mechanism makes one binary attribute of the user differentially private. In this setting, the user can answer any question that requires a binary answer. For example, to answer the question "Are you a non-profit?" the user can reply "0" for non-profit or "1" for pro-profit. To answer the question "Are you based in the U.S.?" the user can reply "0" for a yes or "1" for a no.

We chose this simple implementation as a proof of concept, but argue that

Algorithm 1 VerRR

Input:

- Verifiable randomness ξ
- A private attribute dpa

Output:

- Two coin toss results coin_1 and coin_2
- A differentially private value $\hat{\mathsf{dpa}}$

```
1: Compute first coin toss
   coin_1 = (ξ mod 4) mod 2
2: Compute second coin toss
   coin_2 = (ξ mod 4)/2
3: if coin_1 = 0 then
4:     dpâ ← dpa
5: else
6:     if coin_2 = 0 then
7:         dpâ ← 1
8:     else
9:         dpâ ← 0
10:    end if
11: end if
12: Output coin_1, coin_2, dpâ
```

this mechanism can be easily replaced by a more sophisticated LDP mechanism, such as one capable of handling histogram queries as in the *Rappor* mechanism used by Google [25]. As evident from our security proofs in Section 4.4, our technique achieves simulation security; therefore, the entire protocol inherits the security of the LDP function.

The VerRR algorithm uses the jointly generated random value ξ as the source of randomness needed to compute the double coin toss coin_1 and coin_2. Based on the results of the coin tosses, the algorithm determines the value of output $\hat{\mathsf{dpa}}$ (the original dpa value, or a random output of "0" or "1").

The pseudocode for the VerRR algorithm is given in Algorithm 1.

4.3 The VDP Transfer

The VDPtransfer algorithm expands the underlying transfer algorithm (e.g., the Pour algorithm used by Zcash [5]) by outputting additional information about the user's attributes. The analyst can later use this information to generate aggregate statistics regarding system users. To preserve privacy and allow plausible deniability, the user applies an LDP mechanism to its attributes, making them differentially private before disclosing them.

On a very high level, the algorithm expands the underlying transfer algorithm of unspent m tokens with randomness seeds $\vec{\rho}$ such that $|\vec{\rho}| = m$ executed by user u as follows:

1. u computes the jointly generated random value ξ out of the verifiable random values ξ_u and ξ_A obtained from executing the BindRandomness protocol.

2. u uses ξ as the source of randomness needed to generate the random values used in the VerRR algorithm.

3. u makes their attribute dpa differentially private and encrypts its value using the analyst's public key pk_A.

4. u computes two zero-knowledge proofs—the *Binding* proof π_ξ, and the *Encrypted VDP* proof π_δ—to prove the connection between the jointly generated random values ξ_u, ξ_A, ξ, the unspent tokens with random seeds

Algorithm 2 VDPtransfer

Input:

- Public parameters pp
- Verifiable randomness ξ_u and its commitment cm_{ξ_u}
- Serial number ν_1
- Verifiable randomness ξ_A
- Signature σ_A for $(cm_{\xi_u}, \nu_1, \xi_A)$
- Underlying transfer parameters: x=$((\theta_i^{old})_{i=1}^m, (\mathsf{addr}_{\mathsf{sk},i}^{old})_{i-1}^m, info$[6]$)$

Output:

- New tokens $(\theta_i^{new})_{i=1}^m$
- Encrypted VDP value δ
- Transfer transaction $\mathsf{tx}_{\mathsf{VDP}}$

1: Compute randomness $\xi = \mathsf{ADD}(\xi_u, \xi_A)$
2: Compute commitment $cm_\xi = \mathsf{COMM}(\xi)$
3: for each $i \in [m]$:
   a: Parse θ_i^{old} as (ρ_i, dpa, $\mathsf{addr}_{\mathsf{pk},i}^{old}$, *)
   b: Parse $\mathsf{addr}_{\mathsf{sk},i}^{old}$ to retrieve $a_{\mathsf{sk},i}^{old}$
   c: Parse $\mathsf{addr}_{\mathsf{pk},i}^{old}$ to retrieve $a_{\mathsf{pk},i}^{old}$
4: Compute $\nu_2 = \mathsf{PRF}_2(\vec{\rho})$ *s.t.* $\vec{\rho} := (\rho_1, \ldots, \rho_m)$
5: Compute $\mathsf{VerRR}(\xi, \mathsf{dpa})$ to retrieve $(\mathsf{coin}_1, \mathsf{coin}_2, \hat{\mathsf{dpa}})$
6: Set $\overrightarrow{w_\xi} = (\xi_u, \xi_A, cm_{\xi_u}, \sigma_A, \xi, \vec{\rho}, \nu_1, (a_{sk,i}^{old})_{i=1}^m)$
7: Set $\overrightarrow{x_\xi} = (cm_\rho, \nu_2, (\mathsf{sn}_i^{old})_{i=1}^m, cm_\xi, \mathsf{pk}_A)$
8: Compute proof $\pi_\xi = \mathsf{Prove}(\mathsf{pk}_\xi, x_\xi, a_\xi)$
9: Encrypt and mask attribute $\delta = \mathsf{Enc}_{\mathsf{pk}_A}(r_u, \hat{\mathsf{dpa}})$
10: Set $\overrightarrow{w_\delta} = (\mathsf{coin}_1, \mathsf{coin}_2, \xi, \mathsf{dpa}, r_u, (a_{pk,i}^{old})_{i=1}^m)$
11: Set $\overrightarrow{x_\delta} = (\mathsf{COMM}_\xi, \delta, \mathsf{pk}_A, \mathsf{pk}_{RA})$
12: Compute proof $\pi_\delta = \mathsf{Prove}(\mathsf{pk}_\delta, x_\delta, a_\delta)$
13: Execute underlying transfer Pour(x)
14: Set $\mathsf{tx}_{\mathsf{VDP}} = (\mathsf{tx}_{\mathsf{Pour}}, \delta, \pi_\xi, \pi_\delta)$
15: **Output** $(\theta_i^{new})_{i=1}^n$, $\mathsf{tx}_{\mathsf{VDP}}$

$\vec{\rho}$, and the attribute dpa that was made differentially private. Additionally, u binds $(\rho)_{i=1}^m$ to $\nu_2 = PRF_2(\vec{\rho})$ and sends the underlying transfer encoding, the proofs ξ_u and ξ_A, and ν_2 to the analyst. Sending to the analyst ν_2 ensures that the user cannot reuse ξ a second time. The analyst is expected to add ν_2 to an accumulator ACC and ensure $\nu_2 \notin ACC$ upon receiving it from a user. Since ν_1 is computed using PRF_1 and ν_2 is computed using PRF_2, and $PRF_1 \neq PRF_2$ then also ν_1 is unlinkable to ν_2.

The pseudocode for the $\mathsf{VDPtransfer}$ algorithm is given in Algorithm 2.

4.3.1 The *Binding* Proof

The proof, made by user u, is defined as follows:

$$\pi_\xi = \left[\begin{array}{l|l} \exists \xi_u, \exists \xi_A, & \mathsf{verify}(\mathsf{pk}_A, \sigma_A, "cm_{\xi_u} || \nu_1 || \xi_A") = 1 \wedge cm_{\xi_u} = \mathsf{COMM}(\xi_u) \wedge \\ \exists cm_{\xi_u}, \exists \sigma_A, & \xi = \mathsf{ADD}(\xi_u, \xi_A) \wedge cm_\xi = \mathsf{COMM}(\xi) \wedge \nu_1 = \mathsf{PRF}_1(\vec{\rho}) \wedge \\ \exists \xi, \exists \vec{\rho}, & \nu_2 = \mathsf{PRF}_2(\vec{\rho}) \wedge cm_\rho = \mathsf{COMM}(\vec{\rho}) \wedge \vec{\rho} \in \mathsf{ASC} \\ \exists \nu_1, \exists (a_{sk,i}^{old})_{i=1}^m) & \bigwedge_{i=1}^m \mathsf{sn}_i^{old} = \mathsf{PRF}_{a_{sk}}(\vec{\rho}[i]) \end{array} \right].$$

Where instances are of the form $\overrightarrow{x_\xi} = (cm_\xi, cm_\rho, \nu_2, (\mathsf{sn}_i^{old})_{i=1}^m, \mathsf{pk}_A)$, and witnesses are of the form $\overrightarrow{w_\xi} = (\xi_u, \xi_A, cm_{\xi_u}, \sigma_A, \xi, \vec{\rho}, \nu_1, (a_{sk,i}^{old})_{i=1}^m)$. We define ASC as an m relation $(n_1, n_2, ...n_m) | n_1 < n_2 < ... < n_m$ (i.e., an m relation where all the elements are smaller than the elements to their right).

An instance $\overrightarrow{x_\xi}$ specifies a commitment for a jointly generated randomness, a commitment for the unspent tokens, a public serial number binding the unspent

[6] *info* represents the rest of the parameters needed for the underlying transfer

tokens to the jointly generated randomness, the serial numbers of m distinct token (computed by the underlying token management system), and the analyst's public key. A witness $\overrightarrow{w_\xi}$ specifies user u's randomness and commitment to it, the analyst's randomness and signature for it, the jointly generated randomness, the seeds for the m distinct unspent tokens, the private serial number binding the unspent tokens to the jointly generated randomness, and the m private addresses of the m unspent tokens.

Given a *Binding* proof instance $\overrightarrow{x_\xi}$, a witness $\overrightarrow{w_\xi}$ is valid for $\overrightarrow{x_\xi}$ if the following statements hold:

1. The signature σ_A created by the analyst is valid, i.e., $\mathsf{verify}(\mathsf{pk}_A, \sigma_A, "cm_{\xi_u}||\nu_1||\xi_A") = 1$
2. The commitment cm_ξ is a valid commitment for randomness ξ, i.e., $cm_\xi = \mathsf{COMM}(\xi)$.
3. The randomness ξ was computed using the user's randomness ξ_u and the analyst's randomness ξ_A, i.e., $\xi = \mathsf{ADD}(\xi_u, \xi_A)$.
4. The vector $\vec{\rho}$ is sorted in ascending order, i.e., $(\rho_1, \rho_2, ..., \rho_m)|\rho_1 < \rho_2 < ... < \rho_m$.
5. The serial numbers ν_1, ν_2 are computed correctly, i.e., $\nu_1 = \mathsf{PRF}_1(\vec{\rho})$ and $\nu_2 = \mathsf{PRF}_2(\vec{\rho})$.
6. The public serial number ν_2 matches a private serial number ν_1, s.t. ν_1 appears in the analyst's accumulator.
7. The commitment cm_u is a valid commitment for the randomness ξ_u generated by the user, i.e., $cm_{\xi_u} = \mathsf{COMM}(\xi_u)$.
8. The commitment cm_A is a valid commitment for the randomness ξ_A generated by the analyst, i.e., $cm_{\xi_A} = \mathsf{COMM}(\xi_A)$.
9. For each $i \in [m]$, the serial number sn_i^{old} of token θ_i^{old} is computed correctly, i.e., $\mathsf{sn}_i^{old} = \mathsf{PRF}_{a_{sk}}(\rho_i)$ s.t. $\rho_i = \vec{\rho}[i]$.

4.3.2 The *Encrypted VDP* Proof

The proof, made by user u, is defined as follows:

$$\pi_\delta = \left[\begin{array}{l|l} \exists \xi, \exists \mathsf{dpa}, & cm_\xi = \mathsf{COMM}(\xi) \wedge \hat{\mathsf{dpa}} = \mathsf{VerRR}(\xi, \mathsf{dpa}) \wedge \\ \exists \mathsf{coin}_1, \exists \mathsf{coin}_2, & \mathsf{coin}_1 = (\xi \bmod 4) \bmod 2 \wedge \mathsf{coin}_2 = (\xi \bmod 4)/2 \wedge \\ \exists r_u, \exists (a_{pk,i}^{old})_{i=1}^n & \delta = \mathsf{Enc}_{\mathsf{pk}_R}(r_u, \hat{\mathsf{dpa}})) \wedge \\ & \mathsf{verify}(\mathsf{pk}_R, \sigma_{\mathsf{dpa}}, "(a_{pk,i})_{i=1}^n || \mathsf{dpa}") = 1 \end{array} \right].$$

Where instances are of the form $\overrightarrow{x_\delta} = (\mathsf{COMM}_\xi, \delta, \mathsf{pk}_A, \mathsf{pk}_{RA})$, and witnesses are of the form $\overrightarrow{w_\delta} = (\mathsf{coin}_1, \mathsf{coin}_2, \xi, \mathsf{dpa}, r_u, (a_{pk,i}^{old})_{i=1}^m)$.

An instance $\overrightarrow{x_\delta}$ specifies a commitment for a jointly generated randomness, the encrypted value of the user's attribute after making it differentially private, the analyst's public key, and the registration authority's public key. A witness $\overrightarrow{w_\delta} = (\xi, \mathsf{coin}_1, \mathsf{coin}_2, \mathsf{dpa}, r_u, (a_{pk,i}^{old})_{i=1}^m)$ consists of the jointly generated randomness and the two coin toss results derived from it, the user's attribute, the user's random scalar used during the ElGamal encryption process, and the m public addresses of the m unspent tokens.

Given an *Encrypted VDP* proof instance $\overrightarrow{x_\xi}$, a witness $\overrightarrow{w_\xi}$ is valid for $\overrightarrow{x_\xi}$ if the following statements hold:

1. The commitment cm_ξ is a valid commitment for randomness ξ, i.e., $cm_\xi = \mathsf{COMM}(\xi)$.
2. The double coin toss results coin_1 and coin_2 are derived from randomness ξ, i.e.,
   $\mathsf{coin}_1 = (\xi \bmod 4) \bmod 2$ and $\mathsf{coin}_2 = (\xi \bmod 4)/2$.
3. The encrypted value δ was computed by encrypting the user's attribute dpa with the analyst's public key pk_A after making it differentially private using coin_1 and coin_2, i.e., $\delta = \mathsf{Enc}_{\mathsf{pk}_R}(\mathsf{LDP}(\mathsf{dpa}, \mathsf{coin}_1, \mathsf{coin}_2))$.
4. The signature created by the registration authority is valid, i.e.,
   $\mathsf{verify}(\mathsf{pk}_R, \sigma_{\mathsf{dpa}}, "(a_{pk,i})_{i=1}^n || \mathsf{dpa}") = 1$.

4.4 Security Analysis

In this section, we prove the security of our scheme. Specifically, we prove that our scheme preserves integrity despite malicious parties and preserves the user's privacy.

4.4.1 Preserving Integrity

In our scheme, the user sends the analyst an encrypted result of computing the VerRR algorithm on their private attribute termed dpa. As the analyst does not see how the user computes the noise, it is crucial that the scheme preserves the integrity of the process in case the user is being dishonest. We prove a stronger result; If either (but not both) of the parties is malicious, the final result of our protocol distributes as an ideal functionality $\mathcal{F}$ for a randomized response differential privacy algorithm. In order to prove integrity, we first define three lemmas:

Lemma 1 *Let $\mathcal{U}$ be the uniform distribution. Unless both*[7] *the user and the analyst are malicious, and if the analyst accepts the proofs accompanying the transaction,* $\mathsf{BindRandomness}$ *(see Figure 2) outputs ξ_A and ξ_u such that $\xi_A + \xi_u \sim \mathcal{U}\left(0, 2^\lambda\right)$.*

Lemma 2 *Denote π_ξ and π_δ to be Non-Interactive Zero-Knowledge proofs accepted by the analyst with a corresponding commitment $COMM_\xi$ to ξ and δ being the claimed result*[8] *of applying VerRR on $\mathsf{dpa} \in \{0,1\}^*$ with some randomness ξ'. Then it holds that: $\xi = \xi'$.*

Lemma 3 *Denote dpa to be the user's attribute and δ the result (claimed by the user) of VerRR on some dpa' accepted by the analyst. Then, it holds that indeed $\mathsf{dpa}' = \mathsf{dpa}$.*

[7] If both are malicious, we cannot guarantee anything about the distribution of the final randomness $\xi = \xi_A + \xi_u$ as ξ_A, ξ_u may both not be sampled uniformly.

[8] In the protocol, δ is an encryption of the result, but for simplicity, we omit this.

The proofs can be found in Appendix B. Now that we have proved the lemmas above, we can prove the integrity theorem (proof found in Appendix B):

Theorem 1 (preserving integrity) *Let $\mathcal{F}$ an ideal functionality for the* VerRR *computation where the user (U) sends its data* dpa *and the analyst (A) receives $\mathcal{F}(\mathsf{dpa})$, and denote $\langle U(\mathsf{dpa}), A\rangle$ as a random variable that represents the final LDP value to be sent to the analyst. Then, unless both the user and analyst are malicious, it holds that: $\langle U(\mathsf{dpa}), A\rangle \sim \mathcal{F}(\mathsf{dpa})$.*

Although Theorem 1 and its auxiliary lemmas are enough to prove the user cannot falsify $\mathcal{F}(dpa)$, a malicious user may attempt to execute several instances of the **BindRandomness** protocol and pick one result over the other in order to skew the randomness in its favor. We, therefore, prove that our scheme has a *finality* property with respect to the execution of **BindRandomness**:

Theorem 2 (Finality) *Let $\{sn_i\}_{i=1}^m$ be a set of serial numbers exposed by the underlying payment scheme, which correspond to tokens being spent in a future transaction by a user, and let $\vec{\rho}$ be its corresponding vector of random seeds. Denote $\{(\xi_u^{(i)}, \vec{\rho}) \mid \xi_u \in \{0,1\}^\lambda\}_{i=1}^\infty$ to be an infinite series of inputs to **BindRandomness** with the aforementioned $\vec{\rho}$ but with a different randomness $\xi_u^{(i)}$ each time. Then, only the first element in the series, $(\xi_u^{(1)}, \vec{\rho})$ grants the user with an output from the Analyst.*

The proof for Theorem 2 can be found in the appendix. Note that the binding proof π_ξ uses as input the serial numbers $\{sn_i\}_{i=1}^m$ and ensures that the input seeds for the serial numbers $\vec{\rho}$ are the same as the input seeds for ν_1 and ν_2, therefore a user cannot pick ν_1' for which $\nu_1' \neq PRF_1(\vec{\rho})$ in **BindRandomness** and then spend tokens corresponding to $\{sn_i\}_{i=1}^m$.

4.4.2 Preserving User Privacy

We show that our scheme does not degrade the privacy of the underlying transfer scheme which is used in a black box manner.

Theorem 3 (preserving privacy) *Let U be an honest user that completed a* VDPtransfer *preceded by interacting with the analyst via* BindRandomness. *Once the transfer completes, the analyst's guess about which user sent the* VDPtransfer *or interacted via* BindRandomness *is the same as it was before.*

In Appendix B, our proof for the theorem shows that the zero-knowledge proofs used in our scheme are simulatable, and hence anything computable by the analyst after observing them could have been computed before observing them.

5 Evaluation and Implementation

We implement our scheme in $\sim$ 500 lines of Go using the Gnark ZK-SNARK library [9]. Our implementation is available publicly in [18]. We use the Groth16 [27] scheme instantiated with the BN-254 curve as it is the most efficient according to the evaluation of [21].

Table 1: Performance of π_ξ proof. Times are in milliseconds

Inputs	Constraints	Setup	Proof	Ver
1	9769	725	99	0.944
2	12678	889	113	0.959
4	18496	1417	164	0.971
8	30132	2064	216	1
16	53404	3741	364	1.04

Table 2: Performance of π_δ proof. Times are in milliseconds

Inputs	Constraints	Setup	Proof	Ver
1	12882	869	103	0.93
2	13155	879	104	0.935
4	13701	907	107	0.941
8	14793	966	114	0.951
16	16977	1288	157	0.961

Our choices of public key encryption, digital signatures, commitment schemes, and Pseudo-Random Functions were influenced by efficiency considerations, particularly the cost of operations in arithmetic circuits over a finite field that is the order of the BN-254 curve.

We use the MiMC [1] hash function for the Pseudo-Random Function and the commitment scheme. Although it gives only computational hiding and not information-theoretic hiding such as the Pedersen [39] commitment, it is cheaper because of the smaller[9] number of constraints.

For Public Key Encryption and digital signatures, we use Elgamal and ed-DSA, respectively, over a curve tailored to be efficient for ZK-SNARKs as it is defined over a field whose order [10] is the order of the BN-254 curve.

We investigate the overhead of using our scheme in conjunction with a privacy-preserving asset transfer by evaluating the time our proofs require.

We benchmark on a c5a.2xlarge AWS machine equipped with 8 vCPUs and 16GB RAM. We evaluate the performance of both our proofs (π_ξ, π_δ) by running 100 independent trials and computing the number of constraints, averages for setup time, proof time, and verification time for different numbers of input tokens. The results are depicted in Table 1 and Table 2 for π_ξ and π_δ respectively.

As seen from the performance evaluation results shown in Figure 3, our scheme has a practical execution time. Moreover, the number of inputs adds a negligible increase to the verification time, and has a small magnitude on the proof generation time. Additionally, as the verification time is very short (totaling less than 2ms for all input counts up to 16 inputs), we conclude that the additional computations and data that we added with our VDPtransfer transaction do not add much overhead to the original underlying transfer.

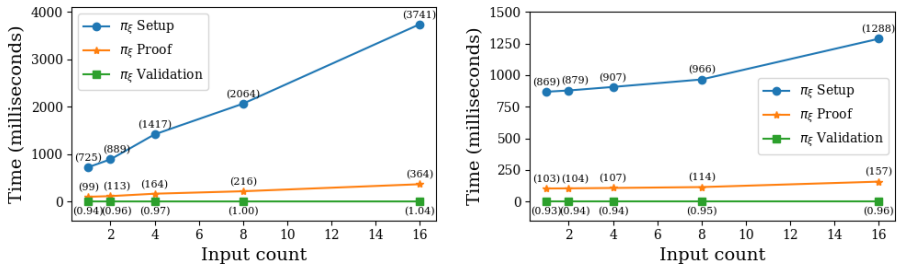
6 Discussion

In this section, we analyze some of our design choices and discuss the implications of our scheme.

6.1 The LDP Mechanism

Embedding any local differential privacy mechanism, especially a verifiable one, into a privacy-preserving transaction system, is a significant challenge. This is due to the need to maintain user privacy and prevent the unintentional leakage of information during data disclosure. For instance, applying the well-known

[9]Scalar multiplication in Elliptic Curves has logarithmic complexity



(a) π_ξ Proof (b) π_δ Proof

Figure 3: Performance Evaluation of both proofs as a function of the number of unspent inputs used in each transaction.

Laplace mechanism [23] to render transaction values differentially private is ineffective when dealing with outliers, such as exceptionally high transaction values [40]. Merely adding noise following a Laplace distribution would not sufficiently conceal these extreme values, thereby compromising the privacy of the user involved in such a transaction. This approach, thus, does not align with our context. Given these observations, our focus has shifted to data requests that necessitate binary responses, which are inherently less susceptible to outlier effects.

For our local differential privacy mechanism, we utilized the randomized response technique [24]. We chose this mechanism for several reasons. Firstly, the randomized response method is based on simple probabilities, making it relatively straightforward to understand and analyze [44]. This characteristic is crucial when deploying the mechanism in a distributed, trustless environment. Secondly, the randomized response is well-suited for handling discrete values, aligning with our scheme's binary attribute. Lastly, the randomized response technique is suitable for local differential privacy and demonstrates strong accuracy and low error bounds, particularly when applied to many users [8]. These attributes make it applicable and well-suited to address the threat models specific to our research.

The challenge in a privacy-preserving transaction system was achieving verifiable randomness without revealing the user's identity. To the best of our knowledge, our approach is pioneering in addressing this issue. We devised a non-interactive method that employs two serial numbers for verification. While the user generates these serial numbers, the data analyst cannot link specific randomness to a particular transaction due to the unique creation and utilization of these numbers. Specifically, the first serial number is exclusively generated and applied during the randomness acquisition, and the second is solely utilized during the transfer as a component of the zk-SNARK proof, as detailed in Section 4.

6.2 The Zero-Knowledge Proof Scheme

Our approach uses the ZK-SNARK scheme of Groth16 [27], necessitating a trusted setup. A Zero-Knowledge proof system with a trusted setup stipulates that the randomness used during the setup process must be discarded to prevent

misuse. If this randomness were to leak, any party possessing it could generate a deceptive proof. Contrarily, in a transparent setup, the randomness is publicly accessible and known to all parties, including the verifier, eliminating the need for an external trusted setup phase. Although it may seem that the transparent setup might be superior, in our specific context, a trusted setup aligns perfectly with our adversarial model, negating the need for a proof scheme incorporating a transparent setup.

The trusted setup is viable for us since our model consists of a single analyst and multiple (potentially infinite) users, where the analyst solely serves as a verifier, and the users operate as provers. Therefore, the analyst can generate the trusted setup since there is no other party that has to verify the proof generated by the analyst. Furthermore, since the data a user sends comprises cryptographic commitments, the analyst cannot derive any information from what a user sends, even if the randomness used during the trusted setup was not discarded.

6.3 Incentivising Conformation by Design

Our scheme has two phases: (a) obtaining randomness; (b) using it in a transaction. When obtaining the randomness to be later used in a transaction, the randomness is only bound to the unspent outputs that the user wishes to spend in the future. Consequently, if the user decides to use an unspent output for a transaction to some recipient, they may change their mind about the recipient but not about the (unspent) input tokens.

At first glance, it may seem like a dishonest user may want to deliberately skew the analyst's statistics by consistently selecting noise that hides their data. Such a dishonest user may obtain randomness for an unspent output, compute the corresponding noise, and if the noise is "bad", they have three strategies: (i) Throw away the randomness and never use it; (ii) Use the randomness by sending its corresponding unspent outputs back to themselves and then repeating the process by obtaining new randomness; (iii) Trying to manipulate the randomness by requesting new randomness corresponding to different subsets of unspent outputs.

Discarding the randomness, as suggested in strategy (i), also means discarding the funds associated with the unspent outputs used to generate the randomness, which incurs a significant cost. Hence, there is an incentive for a dishonest user not to do so.

Surprisingly, using randomness by sending the unspent outputs back to the user, as suggested in strategy (ii), does not affect the analyst, as the analyst cannot differentiate between a transfer to the same user and a transfer to a different user in the first place. In other words, the analyst's aggregated statistics stay the same whether dishonest users pick this strategy or not.

This leaves us with the dishonest user's last strategy (i.e., strategy (iii)), binding various combinations of unspent outputs with the same total sum for one of the combinations to yield noise that hides the user's data. In our implementation, the user cannot simply reorder the set of unspent inputs and request a new randomness since, as part of the π_ξ proof, the user proves that the random seeds of the unspent inputs are sorted in ascending order. Therefore, the only reason strategy (iii) is possible is that in our implementation, BindRandomness computes ν_1 as a PRF on a *vector* $\omega = (\rho_1, \rho_2, ..., \rho_m)$. In-

deed, if the user has unspent outputs corresponding to $\{\rho_1, \rho_2, ..., \rho_l\}$ such that $l > m$, it has $\binom{l}{m}$ independent attempts of obtaining a randomness it desires. However, such a strategy can be easily mitigated by defining ν_1 as a *vector* instead of a single value. Specifically, if in BindRandomness the user sends: $\vec{\nu_1} = (PRF_1(\rho_1), PRF_1(\rho_2), ..., PRF_1(\rho_m))$ and the analyst checks $\nu_{1,i} \notin ACC$ for every $\nu_{1,i} \in \vec{\nu_1}$. This strategy becomes equivalent to the aforementioned strategy (i). We note that defining ν_1 as a *vector* instead of a single value also eliminates the need for the set of unspent inputs to be in ascending order.

7 Conclusions

In this work, we describe the VDP transaction scheme that fits the needs of digital payment systems that require built-in governance and regulations such as CBDCs. The scheme combines privacy-preserving transactions with statistical insights gathering without harming privacy. Since the VDP transaction scheme expands the functionality of any given privacy-preserving transaction system, it can uphold privacy guarantees towards the users. At the same time, since the VDP transaction scheme incorporates a mechanism for verifiable LDP, it can provide users with plausible deniability and prevent bias in users' responses, thus maintaining the integrity of the statistical insights. To achieve verifiability, we adapt the implementation of the *random response* mechanism; We replace the randomness used in the original *random response* with a jointly generated randomness and add zk-SNARK proofs. Furthermore, we prove that our scheme can preserve user privacy and statistical insight integrity even if one of the main participants (i.e., the user or the analyst) is malicious.

References

[1] Martin Albrecht, Lorenzo Grassi, Christian Rechberger, Arnab Roy, and Tyge Tiessen. Mimc: Efficient encryption and cryptographic hashing with minimal multiplicative complexity. Cryptology ePrint Archive, Paper 2016/492, 2016. `https://eprint.iacr.org/2016/492`.

[2] Andris Ambainis, Markus Jakobsson, and Helger Lipmaa. Cryptographic randomized response techniques. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography - PKC 2004, 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, March 1-4, 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 425–438. Springer, 2004.

[3] Elli Androulaki, Jan Camenisch, Angelo De Caro, Maria Dubovitskaya, Kaoutar Elkhiyaoui, and Björn Tackmann. Privacy-preserving auditable token payments in a permissioned blockchain system. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, AFT '20, page 255–267, New York, NY, USA, 2020. Association for Computing Machinery.

[4] Elli Androulaki, Jan Camenisch, Angelo De Caro, Maria Dubovitskaya, Kaoutar Elkhiyaoui, and Björn Tackmann. Privacy-preserving auditable token payments in a permissioned blockchain system. In *AFT '20: 2nd*

ACM Conference on Advances in Financial Technologies, New York, NY, USA, October 21-23, 2020, pages 255–267. ACM, 2020.

[5] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 459–474. IEEE Computer Society, 2014.

[6] Alex Biryukov and Sergei Tikhomirov. Deanonymization and linkability of cryptocurrency transactions based on network analysis. In *2019 IEEE European symposium on security and privacy (EuroS&P)*, pages 172–184. IEEE, 2019.

[7] Ari Biswas and Graham Cormode. Verifiable differential privacy for when the curious become dishonest, 2022.

[8] Graeme Blair, Kosuke Imai, and Yang-Yang Zhou. Design and analysis of the randomized response technique. *Journal of the American Statistical Association*, 110(511):1304–1319, 2015.

[9] Gautam Botrel, Thomas Piellard, Youssef El Housni, Ivo Kubjas, and Arya Tabaie. Consensys/gnark: v0.6.4, February 2022.

[10] Reinier Broker. Constructing elliptic curves of prescribed order. June 2006. Retrieved from `https://hdl.handle.net/1887/4425`.

[11] Benedikt Bünz, Shashank Agrawal, Mahdi Zamani, and Dan Boneh. *Zether: Towards Privacy in a Smart Contract World*, pages 423–443. 07 2020.

[12] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Data poisoning attacks to local differential privacy protocols. *CoRR*, abs/1911.02046, 2019.

[13] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. Data poisoning attacks to local differential privacy protocols. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 947–964, 2021.

[14] Albert Cheu, Adam Smith, and Jonathan Ullman. Manipulation attacks in local differential privacy. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 883–900. IEEE, 2021.

[15] Albert Cheu, Adam D. Smith, and Jonathan R. Ullman. Manipulation attacks in local differential privacy. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 883–900. IEEE, 2021.

[16] Christian Covington, Xi He, James Honaker, and Gautam Kamath. Unbiased statistical estimation and valid confidence intervals under differential privacy. *arXiv preprint arXiv:2110.14465*, 2021.

[17] Ana-Maria Creţu, Federico Monti, Stefano Marrone, Xiaowen Dong, Michael Bronstein, and Yves-Alexandre de Montjoye. Interaction data are identifiable even across long periods of time. *Nature Communications*, 13(1):313, 2022.

[18] Danielle Movsowitz Davidow, Yavoc Manevich, and Eran Toch. Verifiable differential privacy, 2022.

[19] Yves-Alexandre De Montjoye, Laura Radaelli, Vivek Kumar Singh, and Alex "Sandy" Pentland. Unique in the shopping mall: On the reidentifiability of credit card metadata. *Science*, 347(6221):536–539, 2015.

[20] Maya Dotan, Saar Tochner, Aviv Zohar, and Yossi Gilad. Twilight: A differentially private payment channel network. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 555–570, 2022.

[21] Guillaume Drevon and Aleksander Kampa. Benchmarking zero-knowledge proofs with isekai, 2019.

[22] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming*, pages 1–12, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[23] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[24] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[25] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067, 2014.

[26] Stephen H Fuller and Ariel Markelevich. Should accountants care about blockchain? *Journal of Corporate Accounting & Finance*, 31(2):34–46, 2020.

[27] Jens Groth. On the size of pairing-based non-interactive arguments. *IACR Cryptol. ePrint Arch.*, page 260, 2016.

[28] Zhangshuang Guan, Zhiguo Wan, Yang Yang, Yan Zhou, and Butian Huang. Blockmaze: An efficient privacy-preserving account-model blockchain based on zk-snarks. *IEEE Transactions on Dependable and Secure Computing*, 19(3):1446–1463, 2022.

[29] Muneeb Ul Hassan, Mubashir Husain Rehmani, and Jinjun Chen. Differential privacy in blockchain technology: A futuristic approach. *Journal of Parallel and Distributed Computing*, 145:50–74, 2020.

[30] Muneeb Ul Hassan, Mubashir Husain Rehmani, and Jinjun Chen. Anomaly detection in blockchain networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 2022.

[31] Wael Issa, Nour Moustafa, Benjamin Turnbull, Nasrin Sohrabi, and Zahir Tari. Blockchain-based federated learning for securing internet of things: A comprehensive survey. *ACM Computing Surveys*, 55(9):1–43, 2023.

[32] Bin Jia, Xiaosong Zhang, Jiewen Liu, Yang Zhang, Ke Huang, and Yongquan Liang. Blockchain-enabled federated learning data protection aggregation scheme with differential privacy and homomorphic encryption in iiot. *IEEE Transactions on Industrial Informatics*, 18(6):4049–4058, 2021.

[33] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. Extremal mechanisms for local differential privacy. *Advances in neural information processing systems*, 27, 2014.

[34] Shiva Prasad Kasiviswanathan, Homin K Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

[35] Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. Preventing manipulation attack in local differential privacy using verifiable randomization mechanism. In Ken Barker and Kambiz Ghazinour, editors, *Data and Applications Security and Privacy XXXV - 35th Annual IFIP WG 11.3 Conference, DBSec 2021, Calgary, Canada, July 19-20, 2021, Proceedings*, volume 12840 of *Lecture Notes in Computer Science*, pages 43–60. Springer, 2021.

[36] Gonzalo Munilla Garrido, Johannes Sedlmeir, and Matthias Babel. Towards verifiable differentially-private polling. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*, pages 1–11, 2022.

[37] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009.

[38] Arjun Narayan, Ariel Feldman, Antonis Papadimitriou, and Andreas Haeberlen. Verifiable differential privacy. In *Proceedings of the Tenth European Conference on Computer Systems*, EuroSys '15, New York, NY, USA, 2015. Association for Computing Machinery.

[39] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 129–140, Berlin, Heidelberg, 1992. Springer Berlin Heidelberg.

[40] Rathindra Sarathy and Krishnamurty Muralidhar. Evaluating laplace noise addition to satisfy differential privacy for numeric data. *Trans. Data Priv.*, 4(1):1–17, 2011.

[41] Georgia Tsaloli and Aikaterini Mitrokotsa. Differential privacy meets verifiable computation: Achieving strong privacy and integrity guarantees. In Mohammad S. Obaidat and Pierangela Samarati, editors, *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications, ICETE 2019 - Volume 2: SECRYPT, Prague, Czech Republic, July 26-28, 2019*, pages 425–430. SciTePress, 2019.

[42] Nicolas Van Saberhagen. Cryptonote v 2.0. 2013. Retrieved from `https://github.com/monero-project/research-lab/blob/master/whitepaper/whitepaper.pdf`.

[43] Yu Wang, Gaopeng Gou, Chang Liu, Mingxin Cui, Zhen Li, and Gang Xiong. Survey of security supervision on blockchain from the perspective of technology. *Journal of Information Security and Applications*, 60:102859, 2021.

[44] Yue Wang, Xintao Wu, and Donghui Hu. Using randomized response for differential privacy preserving data collection. In *EDBT/ICDT Workshops*, volume 1558, pages 0090–6778, 2016.

[45] Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

[46] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

A Proof of $ln3$-Differential Privacy for the Randomized Response Algorithm

Proof 1 *Fix a response. A case analysis shows that*

$$Pr\left[Response = Yes | Truth = Yes\right] = 0.75$$

Specifically, when the truth is "Yes" the outcome will be "Yes" if the first coin comes up tails (probability 0.5), or if the first and second coins come up heads (probability 0.5). Similarly,

$$Pr\left[Response = Yes | Truth = No\right] = 0.25$$

This occurs when the first and second coins come up heads (probability 0.25). Similar reasoning can be applied in the case of the response being "No". Therefore, we obtain:

$$\frac{Pr\left[Response = Yes | Truth = Yes\right]}{Pr\left[Response = Yes | Truth = No\right]} = \frac{0.75}{0.25} = \frac{Pr\left[Response = No | Truth = No\right]}{Pr\left[Response = No | Truth = Yes\right]} = 3$$

B Security Analysis

Proof 2 (Proof of Lemma 1) *We split the proof into two cases:*

1. *The user is malicious. If the user is malicious, then the analyst is honest. Therefore, ξ_A is sampled from a uniform distribution. If ξ was computed correctly by the user then $\xi \sim \mathcal{U}\left(0, 2^{\lambda}\right)$ because $\xi = \xi_A + \xi_u$ and $\xi_A \sim \mathcal{U}\left(0, 2^{\lambda}\right)$. Otherwise, ξ was not computed correctly, and the analyst rejects the proof π_{ξ}.*

2. *The analyst is malicious. If the analyst is malicious, then the user is honest. If the user aborts in* BindRandomness*, it is because the analyst sent a malformed σ_A. Therefore the lemma holds vacuously (the analyst did not accept the proofs because the user formed no proofs). Otherwise, the user did not abort in* BindRandomness *and proceeded to compute the joint randomness ξ. It holds that $\xi \sim \mathcal{U}\left(0, 2^{\lambda}\right)$ because $\xi_u \sim \mathcal{U}\left(0, 2^{\lambda}\right)$.*

Proof 3 (Proof of Lemma 2) *Assume in contradiction that $\xi' \neq \xi$. Since the analyst accepted π_ξ, it holds that $\exists \sigma_A$ such that*

$$verify\left(pk_A, \sigma_A, "cm_{\xi_u} || \nu_1 || \xi_A"\right) = 1.$$

It follows from the collision resistance property of the commitment scheme that $\exists \xi_u$ and only one such that ξ_u was committed to by the user in BindRandomness*. Since the user and the analyst cannot be both malicious, there exists only a single ξ for which π_ξ is valid; therefore, a single corresponding $COMM_\xi$ exists. However, from the definition of π_δ, it operates on a commitment with the same value of $COMM_\xi$, however $\xi' \neq \xi$ in contradiction to the collision resistance property of the commitment scheme.*

Proof 4 (Proof of Lemma 3) *By definition of π_δ, there exists a signature σ_{dpa} on* dpa *that is verifiable under the registration authority's public key pk_R. Furthermore, by the definition of π_δ, it holds that δ is computed on the same* dpa*, therefore* $\mathsf{dpa}' = \mathsf{dpa}$.

Proof 5 (Proof of Theorem 1(Preserving integrity)) *Follows from the conjunction of lemmas 1 to 3: From Lemma 1, it follows that the combined randomness (later to be termed ξ) distributes uniformly. This randomness is then verified to be used by Lemma 2, and Lemma 3 ensures that the input to the* VerRR *computation is the user's data and not someone else's. It follows from the correctness of the three lemmas above that the user computed the* VerRR *algorithm on her own data using a uniformly random variable sampling. Therefore the result has the same distribution as the result from the ideal functionality $\mathcal{F}$.*

Proof 6 (Proof of Theorem 2(Finality)) *It follows from the construction of the* ***BindRandomness*** *protocol (see Fig. 2) that the first element in the series $\xi_u^{(1)}$ to be used as input will cause the Analyst to receive the tuple $cm_{\xi_u^{(1)}}, \nu_1$. Following lines 5-6 of the protocol, ν_1 is added to the accumulator of the Analyst, and therefore all the subsequent elements in the input series $\xi_u^{(2)}, \xi_u^{(3)}, ...$ will cause the Analyst to abort, as all corresponding commitments $cm_{\xi_u^{(2)}}, cm_{\xi_u^{(3)}}, ...$ are accompanied with ν_1 which was added to the accumulator of the Analyst.*

Proof 7 (Proof of Theorem 3(Preserving privacy)) *Since* VDPtransfer*'s output is two zero-knowledge proofs π_δ and π_ξ, there exist two simulators $\mathcal{S}_\delta$ and $\mathcal{S}_\xi$ that output a transcript that is indistinguishable to the analyst from a real encoding of π_δ and π_ξ. It remains to argue about the preceding interactive step where the user obtains her randomness from the analyst. We construct a simulator $\mathcal{S}$ for* BindRandomness *and the* VDPtransfer *that outputs a transcript that is indistinguishable from the analyst's view. The simulator $\mathcal{S}$ samples $\vec{\rho}'$ uniformly at random from the domain of the underlying token transfer scheme, derives ν_1' and commits to it forming cm_{ξ_u} as the real user does and sends it to*

the analyst. Next, if the analyst properly produces σ_A, the simulator continues. Otherwise, it aborts. If the simulator $\mathcal{S}$ continues, then it calls $\mathcal{S}_\delta$ and $\mathcal{S}_\xi$ and outputs their transcripts. Under the assumption that an honest user also aborts if it detects the analyst sending it a malformed signature σ_A, it can be seen that $\mathcal{S}_\delta$ and $\mathcal{S}_\xi$ only run if σ_A is well formed and valid. It remains to be argued that the joint distribution of all messages sent by the simulators $\mathcal{S}_\delta$, $\mathcal{S}_\xi$ and $\mathcal{S}$ is indistinguishable from the joint distribution in the real case. Clearly, it is not identical, as the simulator $\mathcal{S}$ does not have $\vec{\rho}$, and the real user does. However, $\vec{\rho}'$ is a vector of a uniformly random element sampled from the same known distribution of $\vec{\rho}$; therefore, $\nu_1' \sim \nu_1$ and $\nu_2' \sim \nu_2$ from the real protocol, and thus the entire transcript is indistinguishable to the analyst from the messages sent in the real protocol.