Privacy-Preserving Payment System With Verifiable Local Differential Privacy

Danielle Movsowitz Davidow*
Tel-Aviv University
Israel
dani.movso@gmail.com

Yacov Manevich
IBM Research
Israel
yacov.manevich@ibm.com

ABSTRACT

In permissioned digital currencies such as Central Bank Digital Currencies (CBDCs), data disclosure is essential for gathering aggregated statistics about the transactions and activities of the users. These statistics are later used to set regulations.

Differential privacy techniques have been proposed to preserve individuals' privacy while still making aggregative statistical analysis possible. Recently, privacy-preserving payment systems fit for CBDCs have been proposed. While preserving the privacy of the sender and recipient, they also prevent any insightful learning from their data. Thus, they are ill-qualified to be incorporated with a system that mandates publishing statistical data.

We show that differential privacy and privacy-preserving payments can coexist even if one of the participating parties (i.e., the user or the data analyst) is malicious. We propose a modular scheme that incorporates verifiable local differential privacy techniques into a privacy-preserving payment system. Thus, although the noise is added directly by the user (i.e., the data subject), we prevent her from manipulating her response and enforce the integrity of the noise generation via a novel protocol.

KEYWORDS

Blockchain, Privacy, Verifiable Differential Privacy

1 INTRODUCTION

Thousands of digital currencies, better known as cryptocurrencies, have emerged over the last decade. Bitcoin is the first digital currency to achieve widespread adoption [24]. These currencies are based on an immutable and distributed ledger known as the *blockchain*. Recently, central banks worldwide have been showing a growing interest in issuing central bank digital currencies (CBDC) [27]. CBDCs are a digital form of a country's fiat money that would be widely available to the general public. CBDCs are designed to be similar to cryptocurrencies.

Most cryptocurrencies are designed for unregulated *permissionless* (i.e., public) blockchains and allow the public to participate without a specific identity. However, financial institutions favor *permissioned* platforms as they require identity management, auditability, and non-deniability to uphold regulations set by the government's monetary policies.

The main objectives of a central bank are to maintain currency stability, support monetary policies set by the government, and support the financial system's stability. Central banks collect and analyze all the data and information they need to achieve their objectives and fulfill their duties. The information collected by the central bank includes, among other things, sensitive personal information (e.g., gender, age, and employment status) and information about transactions (e.g., transaction type, monetary value, and participating parties). To increase transparency and economic efficiency and promote economic research, central banks make their statistics and supporting data available to the general public and market participants. Since the central bank is an official institution of a country, it can collect data in its raw format without privacy restrictions. To maintain confidentiality and information security before publishing the data, they anonymize it using different techniques such as Differential Privacy (DP).

Recently, privacy-preserving payment systems fit for CBDCs have been proposed [3]. Such systems produce transactions that are indistinguishable from random, and as a result, nothing can be learned from observing them. Hence, a privacy-preserving payment system prevents any insight gathering or learning from the data. Therefore, using such systems hinders the ability of central banks to collect the data they need. To mitigate the data collection problem, central banks would need access to the original transaction information, or they would need to rely on user cooperation while collecting the necessary data. As users are inclined to keep their attributes and transactions private, central banks cannot fully rely on their honest cooperation. Dishonest users can affect the integrity of the data collected, for example, by manipulating their responses or adding noise to their data in a biased method [10, 13].

We note that privacy-preserving data disclosure in not required only in the case of CBDCs, but in all token systems that require built-in governance and regulations such as enterprise networks [26].

To bridge the gap between preserving user privacy and upholding data integrity, we propose the Verifiable Differentially Private (VDP) transaction scheme. This scheme incorporates *verifiable* local differential privacy (LDP) based on zero-knowledge proofs into a privacy-preserving payment system. By incorporating LDP techniques, institutions (such as central banks) that need aggregated statistics about their users can require those users to disclose their private data while preserving user privacy and allowing plausible deniability for the users. By adding zero-knowledge proofs to the LDP protocol, those institutions can verify the integrity of the data disclosed.

LDP mechanisms are usually based on a user sampling some randomness and then using it for noise generation. This randomness must be kept secret from anyone but the user to ensure the user's privacy. At the same time, to ensure the integrity of the data, the mechanism should force the user to generate and apply the randomness in a non-biased manner. Therefore, to achieve these requirements, even if one of the main participants (i.e., the user or the analyst) is malicious, an LDP mechanism must uphold the following

*The work was done during an internship at IBM research.

properties: (i) Neither the user nor the analyst can bias the generation of the noise; (ii) The user can prove in a privacy-preserving manner that the noise is non-biased; (iii) Once the randomness for a specific input is computed, the user cannot compute new randomness or add a different amount of noise. We present the VerRR algorithm as a simple, verifiable LDP mechanism based on *randomized response* [17] that upholds the properties mentioned above. The VerRR algorithm makes one binary attribute of a user (e.g., gender, account type) differentially private.

An essential aspect of our VDP transaction scheme is its modularity; Each component can be changed or adjusted as needed, and the scheme is not tied to any specific embodiment of any of its components. Our main contributions are:

- The BindRandomness protocol. This protocol is designed to agree on a non-biased randomness to be used in a probabilistic algorithm while the randomness is verifiable but at the same time privacy-preserving.
- The VDPtransfer scheme, which incorporates the VerRR algorithm. This scheme provides any graph-hiding privacy-preserving payment system with the ability to disclose the private attributes of its users by applying a verifiable LDP technique without harming their privacy.

The remainder of the paper is structured as follows. In Section 2, we provide the relevant background on blockchain-based privacy-preserving payment systems and verifiable differential privacy. In Section 3, we introduce some preliminary concepts. We present a high-level overview of our scheme in Section 4, and dive into the scheme's full details in Section 5. In Section 6, we evaluate the performance of our scheme. We discuss design choices in Section 7 and summarize our contribution in Section 8.

2 BACKGROUND

This section gives a layout of the existing state-of-the-art on verifiable differential privacy and briefly summarizes how privacy-preserving payment systems operate in blockchains.

2.1 Blockchain-Based Privacy-Preserving Payment Systems

When modeling tokens or coins, there are two main approaches: (i) The account model; (ii) The Unspent Transaction Output (UTXO) model. Each model has its benefits. The account model is more user-friendly as it presents an abstraction of accounts and balances, and it is the model used in Ethereum [32]. The UTXO model is used in Bitcoin [24] and zcash [6] and presents an abstraction of a plurality of coins. Each coin has a balance and can be spent by fulfilling a condition encoded in it (in most cases, presenting a signature that is verifiable under a public key that its hash is encoded in the coin). The advantage of the UTXO model over the account model is that it is more concurrent when used in a privacy-preserving setting. While it is considered an open problem how to "hide" the sender among all possible senders in an account model [20], "hiding" the sender in the UTXO model is easy. Therefore, in this work, we focus on the UTXO model.

We describe the basics behind a privacy-preserving payment system in the UTXO model. In this model, when a sender transfers funds to a recipient, to prevent leakage of sensitive information that may identify the parties in the payment, three things need to be hidden from prying eyes: (i) the sender's identity, (ii) the recipient's identity, and (iii) the amount transferred. Hiding the transferred amount is usually done by encoding the amount in a cryptographic commitment and providing a zero-knowledge proof that the sum of input coins is greater or equal to the sum of the output coins. Hiding the recipient is done by creating outputs that either have one-time public keys [30] or by hiding the output token's identity by having the coin itself be a commitment [6] to its properties (e.g., owner, amount). We are left with hiding the sender; this is where the various techniques vary the most. In Monero [30], the source address is hidden among a randomly picked set of potential source addresses via a ring signature. In Zcash [6], all coins are leaves in a Merkle tree maintained by the network participants, and by attaching a zero-knowledge proof of membership in the Merkle tree of the spent token, it is guaranteed that the coin exists even though it is impossible to know which exact coin was spent. In order to ensure the coin cannot be spent more than once, Zcash forces the transaction to include a deterministic tag (termed 'nullifier' or 'serial number') that uniquely identifies the coin yet prevents identifying the coin by the serial number. This is done by including some randomness in the input to the commitment that forms the coin, and defining the serial number as an application of a one-way function on that randomness.

In our work, we focus on payment systems that adopt the approach of Zcash [6] for hiding the sender, recipient, and amount. We mention the work of [4], which suggests a scheme similar to Zcash but more fit for CBDC and permissioned settings, and replaces the Merkle tree membership proof with proof of knowledge of a signature of an entity that certifies the coin. Throughout the paper, we use Zcash terminology. Nevertheless, we note that our scheme is flexible enough to be used with any privacy-preserving payment system that encodes the entire token as a commitment and uses serial numbers as in [4, 6].

2.2 Differential Privacy

Differential privacy (DP) [15, 16] is a formal notion of privacy designed to allow learning useful information about a population while learning nothing 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 introduce randomness to the data.

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 has access to the raw data. Each user sends her data to the curator without noise in this model. Since we assume that analysts requesting access to this dataset are malicious, the curator is in charge of correctly executing the differentially private mechanisms the analysts specify.

Definition 1 ($\epsilon-$differentially privacy [15]). *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 $\mathcal{K}$.

The local model is based on the *randomized response* mechanism as was first proposed by Warner in 1965 [31]. It was formalized in the context of learning by Kasiviswanathan et al. [21]. 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 curator. Making the data differentially private before disclosing it ensures 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 [17]). *An $\epsilon-$local randomized $R : \mathcal{X} \rightarrow W$ is an $\epsilon-$differentially private algorithm that takes as input a database of size $n = 1$.*

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

2.3 Verifiable Differential Privacy

When applying DP techniques to preserve privacy, the standard approach assumes that the noise is produced and applied in an honest manner. However, recent works such as [11, 12] have shown that this assumption is naive and that DP is indeed vulnerable to different manipulation attacks affecting the integrity of the analyzed data.

Several works have proposed different techniques to prevent different types of manipulation attacks. The main objective of all these works is to apply the noise and generate it in a verifiable manner. In the centralized DP model, works such as [7, 25, 29] have focused on a family of use cases in which the data subjects send their data either in a clear or in an encrypted or secret-shared form to a party or a set of parties that jointly form an entity called a "curator". The latter applies noise and outputs an anonymized and privacy-preserving data set for post-processing. While in the LDP model, works such as [2, 22] have focused on adjusting known LDP mechanisms by making them verifiable, thus giving the analyst the ability to verify the process in which data subjects add the noise.

We outline the different approaches for verifiable differential privacy employed by various works and highlight the differences from our technique.

2.3.1 Secure Multi-Party Computation (sMPC). When using sMPC, a data subject secretly shares its vote and distributes the shares across several servers. 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, as we assume the analyst, who is also the curator, can potentially be malicious and is not split into independent parties.

2.3.2 Zero-Knowledge proofs. In [25], the curator assembles a database out of the data elements belonging to different data subjects, and then publishes a cryptographic commitment to the entire database. Upon reception of a query from an entity interested in some statistical property, the curator runs the differential privacy function and additionally produces[1] a NIZK (Non-Interactive Zero-Knowledge proof) that the result corresponds with the aforementioned commitment, and sends both the result and the proof back to the requesting entity. The latter verifies the NIZK against the commitment and determines if the differential privacy mechanism was computed correctly.

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 data subjects 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 data subject itself (unlike the work of [29] where the curator randomly samples the randomness), thus producing noise that is non-biased but also verifiable.

2.3.3 Adjusting known LDP Mechanisms. In [22], Kato et al. leverage cryptographic randomized response techniques [2] to make existing LDP mechanisms verifiable. The most significant disadvantage of their proposed method is that it is interactive. In contrast, our proposed scheme is non-interactive. Interactive methods tend to be more expensive and slower than non-interactive methods as they require multiple round trips and direct communication with the analyst. In addition, unlike non-interactive methods, interactive methods do not allow throttling. Therefore, in a peak of usage in the network, while non-interactive systems will form a queue that will eventually be cleared without the users knowing about the queue, interactive systems may freeze and force the users to wait an unknown period of time.

In [23], 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. Although their approach is similar to the approach we propose in this paper, their technique does not uphold the unlinkability and untraceability properties needed in a privacy-preserving payment system. These properties are not upheld since the prover (i.e., the data subject) signs the randomness used as the base of the LDP mechanism with its private key, and the verifier (i.e., the analyst) needs to know the prover's public key to verify the integrity of the response. By knowing the prover's public key, the analyst can later make a connection between the generated randomness and the transfer it was used in, thus revealing the identity of the user making the transfer.

3 PRELIMINARIES

We recall key cryptographic and privacy building blocks.

3.1 Randomized Response

Dwork and Roth presented in [17] a variant of the randomized response mechanism, in which a user answers a "Yes" or "No" question as follows:

[1]For simplicity of presentation, we state that the curator produces the NIZK, but in the cited paper it is delegated to a different party which is an analyst.

(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.

The randomization in this algorithm comes from flipping two coins and creates uncertainty about the true answer. This uncertainty is the source of privacy, as it gives plausible deniability to the data subject.

We prove that the randomized response algorithm described above satisfies $\ln 3$–differentially privacy.

Proof. 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 comes 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 to the case of the response being "No". Therefore, we obtain:

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

□

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.

3.2 Basic Cryptographic Building Blocks

In this section, we outline self-contained building blocks that we use in our scheme. As zero-knowledge proofs play a central part in our scheme, we elaborate on them in the following section. We use λ to denote a system-wide security parameter.

3.2.1 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[2] function ϵ such that for every two messages m_0, m_1 chosen by $\mathcal{A}$ and a commitment $cm \leftarrow Commit\left(m_b\right)$ for $b \in \{0,1\}$ it holds that:

$$Pr\left[b \leftarrow \mathcal{A}\left(pp, cm\right)\right] - \frac{1}{2} \leq \epsilon\left(\lambda\right).$$

- **Binding**: For any probabilistic polynomial adversary $\mathcal{A}$ there exists a negligible function ϵ such that for every m and $cmt \leftarrow Commit\left(m\right)$ and every $m' \neq m$ chosen by $\mathcal{A}$:

$$Pr\left[Open\left(pp, cmt, m\right) = m'\right] \leq \epsilon\left(\lambda\right).$$

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.

3.2.2 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, $Sign\left(m, sk\right)$ outputs a signature σ. For a message m, a public key pk and a signature σ, $Verify\left(pk, m, \sigma\right)$ 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 O which replies to queries denoted by a set Q there exists a negligible function ϵ such that

$$Pr\left[Verify\left(pk, m, \sigma\right) = 1 \wedge m \notin Q\right] \leq \epsilon\left(\lambda\right).$$

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

3.2.3 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, $Enc\left(m, pk\right)$ outputs a ciphertext c. Given a ciphertext c and a private key sk, $Dec\left(c, sk\right)$ outputs a message m. For every public key pk with a corresponding sk and message m it holds that $Pr\left[Dec\left(Enc\left(m, pk\right), sk\right) = m\right] = 1$. 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:

$$Pr\left[\mathcal{A}\left(Enc\left(m, pk\right)\right) = 1\right] - Pr\left[\mathcal{A}\left(Enc\left(m', pk\right)\right) = 1\right] \leq \epsilon\left(\lambda\right).$$

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

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

3.3 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 $x \in \mathcal{L}$:

$$View^P_{V^*}(x) \equiv S(x).$$

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 collision resistant one way function.

At first glance, proving knowledge of a hash pre-image of a string may not seem particularly useful. However, it is vital in higher-level constructions, such as anonymous set membership. Indeed, 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 [6]. In Zcash, a sender wishing to hide the payment source simply creates a zero-knowledge proof that she uses 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[3].

4 OVERVIEW OF OUR SCHEME

This section gives a high-level introduction to our scheme and the entities participating in it.

Our privacy-preserving, verifiable, and differentially-private transfer scheme expands the functionality of any given privacy-preserving payment system (e.g., Zcash [6]) by enabling a third party (e.g., an analyst) to collect statistics about user attributes while preserving user privacy.

Before we elaborate on the participants, components, and transaction flow, we reiterate the scheme's primary objective. The two main actors in our setting are users who potentially have sensitive information they want to keep private and an analyst who aims to collect statistical insights by aggregating queries 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 amount transferred. Each transaction performed by a user is accompanied by an additional result of applying an LDP algorithm[4] to the user's data.

We model our scheme as a protocol between two potentially dishonest parties with conflicting goals. The first is the user who wants to protect her privacy even at the expense of the analyst not learning any useful statistical information. The second is the analyst who wants to collect aggregative statistical information about the user, even if it means de-anonymizing the user's transactions to maintain the collected data's integrity and accuracy. Neither party knows if the counter-party is honest or not, and therefore our protocol needs to ensure the interest of any of the two parties as long as that party is honest. Therefore, a setting where both parties are malicious is outside of our scope.

A key observation is that an LDP mechanism involves sampling randomness and using it for noise generation. Clearly, the randomness must be kept secret from anyone but the user itself, lest a curious analyst can peel off the noise. 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 protocol with the following properties: (i) Neither the user nor the analyst can bias the generation of the noise used for LDP; (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 protocol fulfills all three aforementioned properties in Section 5.4.1.

Another important property of our protocol is that it is privacy-preserving. We prove this property in Section 5.4.2.

4.1 Participants

Our scheme involves the following participants:

Users. They own tokens that represent assets in the real world. Using a transfer transaction, users can exchange tokens within the system. Collecting up-to-date statistics about the users' private attributes (e.g., age, gender, account type) is of interest to third-party entities (e.g., the analyst).

Analyst. This entity has the authority to collect aggregated statistics about the private attributes and activities of the users in the

[3] Zero Knowledge Succinct Non-interactive ARgument of Knowledge

[4] 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.

system. We assume that the analyst is only interested in analyzing statistical information 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 signature) that binds the user's public key to its attributes. The same attributes will later be used as input to the LDP algorithm.

4.2 Components

Our scheme consists of 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 run 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 3.1 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 payment system. As we explain in Section 2, 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 $PRF(\rho)$).

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 first serial number is the serial number created by the user 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 thanks to the fact that both serial 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.

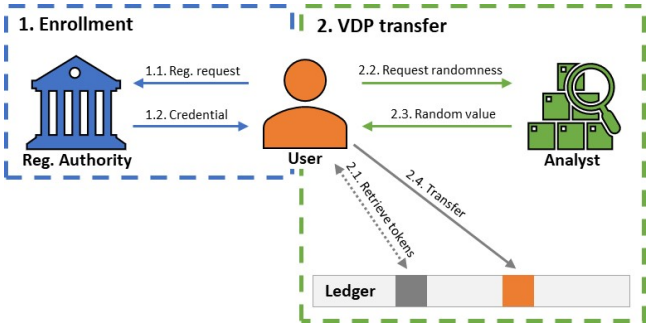


Figure 1: Overview of the VDP Transfer Transaction

Additionally, our transfer uses zk-SNARK proofs to verify the integrity of the data disclosed by the user. The proofs prove that the disclosed data 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.

4.3 VDP Transaction Flow

The VDP transaction flow is illustrated in Figure 1. At first, the user contacts the registration authority and engages in a registration protocol to get a long-term credential. From then on, the user can use this credential to verify its private attributes in every VDP transfer transaction.

The VDP transfer transaction comprises the following stages: (1) The user retrieves its tokens from the ledger. (2) The user contacts the analyst and executes the BindRandomness protocol. Thus, the user obtains a verifiable random value. (3) The user executes the VDP transfer algorithm in three stages. First, the user makes its attributes differentially private and encrypts the result using the analyst's public key. Then, the user computes the zk-SNARK proofs needed to verify the jointly generated randomness and the integrity of the differentially private attribute. Finally, the user executes the underlying transfer transaction. (4) The user submits the encrypted LDP result, and the result of the transaction execution for inclusion in the ledger.

5 THE VDP TRANSACTION SCHEME

This section details the complete VDP transaction scheme.

5.1 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 it knows a random value jointly generated with the analyst so that the random value can be used as a source of randomness for a randomized algorithm.

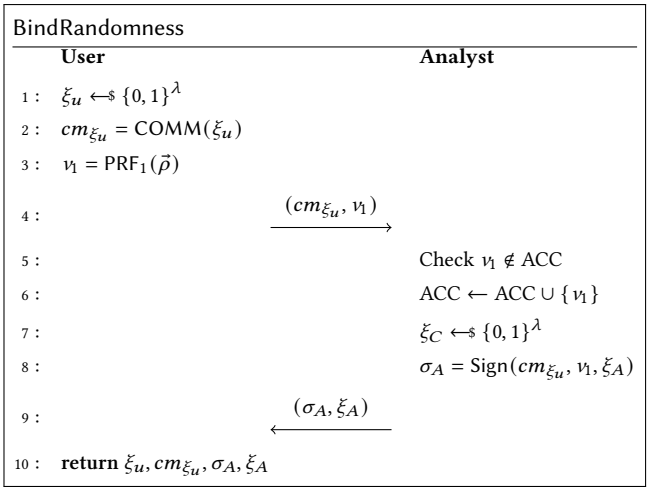


Figure 2: The BindRandomness **protocol.**

The BindRandomness protocol takes as input a security parameter λ and a vector $\vec{\rho} = (\rho_1, \ldots, \rho_m)$ s.t. $\forall i \in [m] : \rho_i \in \{0,1\}^\lambda$. $\vec{\rho}$ 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 in its accumulator a value ν_1 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 see 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.

5.2 The VerRR Mechanism

The VerRR algorithm is a simple, verifiable LDP mechanism based on *randomized response* presented in Section 3.1. 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 "what is your gender?" the user can reply "0" for male or "1" for female. To answer the question "What is the account type?" the user can reply "0" for a private account or "1" for a business account.

We chose this simple implementation as a proof of concept, but argue that 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 [18]. As evident from our security proofs in Section 5.4, our technique achieves simulation security and 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 $\hat{id}$ (the original *id* value, or a random output of "0" or "1").

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

Algorithm 1 VerRR

Input:
- Verifiable randomness ξ
- A private attribute *id*

Output:
- Two coin toss results coin_1 and coin_2
- A differentially private value $\hat{id}$

Compute first coin toss $\mathsf{coin}_1 = (\xi \bmod 4) \bmod 2$
Compute second coin toss $\mathsf{coin}_2 = (\xi \bmod 4)/2$
if $\mathsf{coin}_1 = 0$ **then**
    $\hat{id} \leftarrow id$
else
    if $\mathsf{coin}_2 = 0$ **then**
        $\hat{id} \leftarrow 1$
    else
        $\hat{id} \leftarrow 0$
    end if
end if
Output $\mathsf{coin}_1, \mathsf{coin}_2, \hat{id}$

5.3 The VDP Transfer

The VDPtransfer algorithm expands the underlying transfer algorithm (e.g., the Pour algorithm used by Zerocash [6]) by outputting additional information about the user's attributes. The analyst can later use this information to generate 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 her attributes *id* differentially private and encrypts their values 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 randomness seeds $\vec{\rho}$, and the attributes *id* that were made differentially private. Additionally, u binds $(\rho)_{i=1}^{m}$ to $v_2 = PRF_2(\vec{\rho})$ and sends the underlying transfer encoding, the proofs ξ_u and ξ_A, and v_2 to the analyst. Sending to the analyst v_2 ensures that the user cannot reuse ξ a second time, and the analyst is expected to add v_2 to an accumulator *ACC* and ensure $v_2 \notin ACC$ upon receiving it from a user. Since v_1 is computed using PRF_1 and v_2 is computed using PRF_2, and $PRF_1 \neq PRF_2$ then also v_1 is unlinkable to v_2.

The pseudocode for the VDPtransfer algorithm is given in Algorithm 2.

Algorithm 2 VDPtransfer

Input:

- Public parameters pp
- Verifiable randomness ξ_u and its commitment cm_{ξ_u}
- Serial number v_1
- Verifiable randomness ξ_A
- Signature σ_A for (cm_{ξ_u}, v_1, ξ_A)
- Underlying transfer parameters: $((\theta_i^{old})_{i=1}^{m}, (\mathsf{addr}_{\mathsf{sk},i}^{old})_{i-1}^{m}, info$[5]$)$

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 $(\rho_i, id, \mathsf{addr}_{\mathsf{pk},i}^{old}, *)$
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 $v_2 = \mathsf{PRF}_2(\vec{\rho})$ *s.t.* $\vec{\rho} := (\rho_1, \ldots, \rho_m)$
5: Compute $\mathsf{VerRR}(\xi, id)$ to retrieve $(\mathsf{coin}_1, \mathsf{coin}_2, \hat{id})$
6: Set $\overrightarrow{w_\xi} = (\xi_u, \xi_A, cm_{\xi_u}, \sigma_A, \xi, \vec{\rho}, v_1, (a_{sk,i}^{old})_{i=1}^{m})$
7: Set $\overrightarrow{x_\xi} = (cm_\rho, v_2, (\mathsf{sn}_i^{old})_{i=1}^{m}, cm_\xi, \mathsf{pk}_A)$
8: Compute proof $\pi_\xi = \mathsf{Prove}(\mathsf{pk}_\xi, x_\xi, a_\xi)$
9: Encrypt and mask attribute $\delta = \mathsf{Enc}_{\mathsf{pk}_A}(r_u, \hat{id})$
10: Set $\overrightarrow{w_\delta} = (\mathsf{coin}_1, \mathsf{coin}_2, \xi, id, r_u, (a_{pk,i}^{old})_{i=1}^{m})$
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** $(\theta_i^{new})_{i=1}^{n}, \mathsf{tx}_{\mathsf{VDP}}$

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

5.3.1 The Binding Proof. The proof, made by user u, is defined as follows:

$$\pi_\xi = \left[\begin{array}{l|l} & \mathsf{verify}(\mathsf{pk}_A, \sigma_A, "cm_{\xi_u} || v_1 || \xi_A") = 1 \wedge \\ \exists \xi_u, \exists \xi_A, & cm_{\xi_u} = \mathsf{COMM}(\xi_u) \wedge \\ \exists cm_{\xi_u}, \exists \sigma_A, & \xi = \mathsf{ADD}(\xi_u, \xi_A) \wedge cm_\xi = \mathsf{COMM}(\xi) \wedge \\ \exists \xi, \exists \vec{\rho}, & v_1 = \mathsf{PRF}_1(\vec{\rho}) \wedge v_2 = \mathsf{PRF}_2(\vec{\rho}) \wedge \\ \exists v_1, \exists (a_{sk,i}^{old})_{i=1}^{m}) & cm_\rho = \mathsf{COMM}(\vec{\rho}) \wedge \vec{\rho} \in \mathsf{ASC} \\ & \wedge_{i=1}^{m} \mathsf{sn}_i^{old} = \mathsf{PRF}_{a_{sk}}(\vec{\rho}[i]) \end{array} \right].$$

Where instances are of the form $\overrightarrow{x_\xi} = (cm_\xi, cm_\rho, v_2, (\mathsf{sn}_i^{old})_{i=1}^{m}, \mathsf{pk}_A)$, and witnesses are of the form $\overrightarrow{w_\xi} = (\xi_u, \xi_A, cm_{\xi_u}, \sigma_A, \xi, \vec{\rho}, v_1, (a_{sk,i}^{old})_{i=1}^{m})$. We define ASC as an m relation $(n_1, n_2, \ldots n_m) | n_1 < n_2 < \ldots < n_m$ (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 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., $\mathsf{verify}(\mathsf{pk}_A, \sigma_A, "cm_{\xi_u} || v_1 || \xi_A") = 1$
(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., $(\rho_1, \rho_2, \ldots, \rho_m) | \rho_1 < \rho_2 < \ldots < \rho_m$.
(5) The serial numbers v_1, v_2 are computed correctly, i.e., $v_1 = \mathsf{PRF}_1(\vec{\rho})$ and $v_2 = \mathsf{PRF}_2(\vec{\rho})$.
(6) The public serial number v_2 matches a private serial number v_1, s.t. v_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. $\rho_i = \vec{\rho}[i]$.

5.3.2 The Encrypted VDP Proof. The proof, made by user u, is defined as follows:

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

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 $\overrightarrow{w_\delta} = (\mathsf{coin}_1, \mathsf{coin}_2, \xi, id, r_u, (a_{pk,i}^{old})_{i=1}^{m})$.

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 $\overrightarrow{w_\delta} = (\xi, \mathsf{coin}_1, \mathsf{coin}_2, id, r_u, (a^{old}_{pk,i})_{i=1}^m)$ consists of the jointly generated randomness and the two coin toss results derived from it, the user's attributes, 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 attributes id with the analyst's public key pk_A after making them differentially private using coin_1 and coin_2, i.e., $\delta = \mathsf{Enc}_{\mathsf{pk}_R}(\mathsf{LDP}(id, \mathsf{coin}_1, \mathsf{coin}_2))$.
(4) The signature created by the registration authority is valid, i.e., $\mathsf{verify}(\mathsf{pk}_R, \sigma_{id}, "(a_{pk,i})_{i=1}^n || id") = 1$.

5.4 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.

5.4.1 Preserving Integrity. In our scheme, the user sends the analyst an encrypted result of computing the VerRR algorithm on her private attributes termed id. 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 prove three lemmas:

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

Proof. 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 $\xi_u \sim \mathcal{U}\left(0, 2^\lambda\right)$.

□

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[6] of applying VerRR on $id \in \{0,1\}^*$ with some randomness ξ'. Then it holds that: $\xi = \xi'$.*

Proof. 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. □

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

Proof. By definition of π_δ, there exists a signature σ_{id} on id 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 id, therefore $id' = id$. □

Now that we have proved the lemmas above, we can prove the integrity theorem:

Theorem 1 (preserving integrity). *Let $\mathcal{F}$ an ideal functionality for the* VerRR *computation where the user (U) sends its data id and the analyst (A) receives $\mathcal{F}(id)$, and denote $\langle U(id), A\rangle$ 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: $\langle U(id), A\rangle \sim \mathcal{F}(id)$.*

Proof. 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 $\mathcal{F}$. □

5.4.2 Preserving User Privacy.

Theorem 2 (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.*

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

Proof of **Theorem 2**. 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. □

6 EVALUATION AND IMPLEMENTATION

We implement our scheme in ~ 500 lines of Go using the Gnark ZK-SNARK library [8]. Our implementation is available publicly in [5]. We use the Groth16 [19] scheme instantiated with the BN-254 curve as it is the most efficient according to the evaluation of [14].

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 [28] commitment, it is cheaper due to the smaller [7] number of constraints.

For Public Key Encryption and digital signatures we use Elgamal and edDSA respectively, over a curve tailored to be efficient for ZK-SNARKs as it is defined over a field whose order [9] is the order of the BN-254 curve.

6.0.1 Performance evaluation. 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 sizes 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.

[7] Scalar multiplication in Elliptic Curves has logarithmic complexity

7 DISCUSSION

This section discusses our design choices and their implications.

7.1 The LDP Mechanism

Although some techniques for verifiable differential privacy have been previously researched, none were suitable for the scenario presented in this paper—incorporating a verifiable LDP technique into a privacy-preserving token management system (see Section 2.3).

Incorporating any LDP mechanism into a privacy-preserving token management system, let alone a verifiable LDP mechanism comes with a significant challenge of ensuring that the users' privacy is preserved and that no additional information is leaked during the data disclosure process. For example, in this scenario, trying to make the value of a transfer differentially private using the known *Laplace mechanism* [16] will not work because of outliers that include very high transfer values. Even if we added noise sampled from a Laplace distribution, it would not be enough to hide the unusually high value; therefore, it would not preserve the privacy of the user performing the transfer and does not fit our scenario. Based on this observation, we decided to focus on requests for data that require a binary answer and that could be made differentially private using the simple *randomized response* mechanism [17], as requests like that are less prone to outliers.

To adjust the *randomized response* mechanism, and make it verifiable, the randomness that the mechanism is based on needs to be generated in a verifiable manner. The need for verifiable randomness in a privacy-preserving token system presented us with another challenge—How can a user obtain verifiable randomness while still keeping her identity private? To the best of our knowledge, our proposed scheme is the first to solve this challenge. We solved this challenge by creating a non-interactive process that leverages two serial numbers. Although the user creates the serial numbers, the analyst cannot connect a specific randomness to a specific transfer. The analyst cannot make this connection because of the way the serial numbers are created and used. The first serial number is created and used *only* during the process of obtaining the randomness, and the second serial number is *only* used during the transfer as part of the zk-SNARK proof (see Section 5).

7.2 The Zero-Knowledge Proof Scheme

We discuss the design choices of our underlying Zero-Knowledge proof scheme and show that it is in the interest of a dishonest user to operate as an honest user.

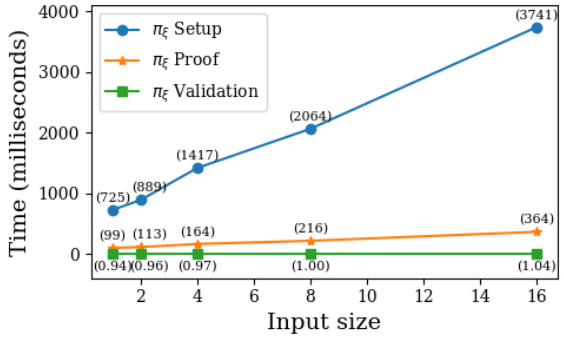
7.2.1 Trusted Setup vs. Transparent Setup. We employ the ZK-SNARK scheme of Goth16 [19], which requires a trusted setup. A Zero-Knowledge proof system with a trusted setup requires that the randomness used during the setup be discarded. Otherwise, if the randomness is leaked to some party, that party may generate proofs on false statements. In comparison, in the transparent setup, the randomness is public and known to all (including the verifier).

Table 1: Performance of π_{ξ} proof. Times are in milliseconds

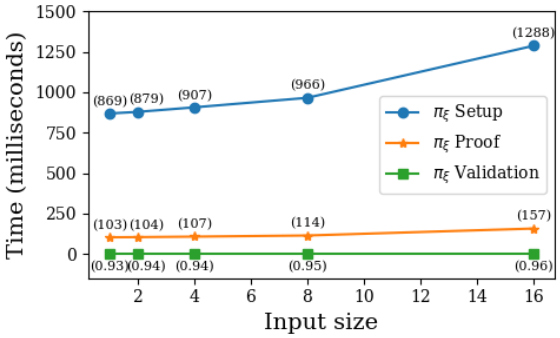
# Inputs	# Constraints	Setup	Proof	Verification
1	9769	725	99	0.944
2	12678	889	113	0.959
4	18496	1417	164	0.971
8	30132	2064	216	1
16	53404	3741	364	1.04

Table 2: Performance of π_{δ} proof. Times are in milliseconds

# Inputs	# Constraints	Setup	Proof	Verification
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



(a) π_{ξ} Proof



(b) π_{δ} Proof

Figure 3: Performance Evaluation

Thus, a transparent setup requires no external trusted setup phase. Although it may seem that the transparent setup is better than the trusted setup, it is important to take into consideration that the transparent setup usually requires a bigger proof size which requires a longer computation time.

Fortunately, in our setting, a trusted setup fits our adversary model perfectly, and there is no need for a proof scheme that has a transparent setup. We can use the trusted setup since our scheme includes a single analyst and many (potentially infinite) users where the analyst acts only as a verifier, and the users act as provers. Therefore, the analyst can generate the trusted setup, as no party needs to verify any proof generated by the analyst. As the data sent by a user is composed of cryptographic commitments, the analyst cannot learn from anything a user sends, even if the randomness used during the trusted setup was never discarded.

7.2.2 Incentivising Conformation by Design. Our scheme has two phases: (a) obtaining randomness; (b) using it in a transfer.

When obtaining the randomness to be later used in a transfer, 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 transfer to some recipient, she may change the recipient right until she actually uses the randomness in a transfer.

At first glance, it may seem like a dishonest user may never reveal her real data or deliberately skew the analyst's statistics by consistently selecting noise that hides her data or has a bias. Such a dishonest user may obtain randomness for an unspent output, compute the corresponding noise, and if the noise is "bad", she has three strategies: (i) To throw away the randomness and never use it; (ii) Use the randomness by sending its corresponding unspent outputs back to herself 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 the randomness by sending the unspent outputs back to the user, as suggested in strategy (ii), has absolutely no effect on 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 do not.

This leaves us with the last strategy (i.e., strategy (iii)) of the dishonest user, which is 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 v_1 as a *PRF* on a *vector* $\omega = (\rho_1, \rho_2, ..., \rho_m)$. Indeed, 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 v_1 as a *vector* instead of a single value. Specifically, if in BindRandomness

the user sends:

$$\vec{v_1} = (PRF_1(\rho_1), PRF_1(\rho_2), ..., PRF_1(\rho_m))$$

and the analyst checks $v_{1,i} \notin ACC$ for every $v_{1,i} \in \vec{v_1}$, this strategy becomes equivalent to the aforementioned strategy (i). We note that defining v_1 as a *vector* instead of a single value also eliminates the need for the set of unspent inputs to be in ascending order.

8 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 payment transfers with statistical insights gathering without harming privacy. Since the VDP transaction scheme expands the functionality of any given privacy-preserving payment 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. 2016. MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity. Cryptology ePrint Archive, Paper 2016/492. https://eprint.iacr.org/2016/492 https://eprint.iacr.org/2016/492.

[2] Andris Ambainis, Markus Jakobsson, and Helger Lipmaa. 2004. Cryptographic Randomized Response Techniques. In *Public Key Cryptography - PKC 2004, 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, March 1-4, 2004 (Lecture Notes in Computer Science, Vol. 2947)*, Feng Bao, Robert H. Deng, and Jianying Zhou (Eds.). Springer, 425–438. https://doi.org/10.1007/978-3-540-24632-9_31

[3] Elli Androulaki, Jan Camenisch, Angelo De Caro, Maria Dubovitskaya, Kaoutar Elkhiyaoui, and Björn Tackmann. 2020. Privacy-Preserving Auditable Token Payments in a Permissioned Blockchain System. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies* (New York, NY, USA) *(AFT '20)*. Association for Computing Machinery, New York, NY, USA, 255–267. https://doi.org/10.1145/3419614.3423259

[4] Elli Androulaki, Jan Camenisch, Angelo De Caro, Maria Dubovitskaya, Kaoutar Elkhiyaoui, and Björn Tackmann. 2020. 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*. ACM, 255–267. https://doi.org/10.1145/3419614.3423259

[5] Anonymous authors. 2022. *Verifiable Differential Privacy*. https://anonymous.4open.science/r/VerifiableDifferentialPrivacy/

[6] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. 2014. Zerocash: Decentralized Anonymous Payments from Bitcoin. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*. IEEE Computer Society, 459–474. https://doi.org/10.1109/SP.2014.36

[7] Ari Biswas and Graham Cormode. 2022. Verifiable Differential Privacy For When The Curious Become Dishonest. arXiv:2208.09011 [cs.CR]

[8] Gautam Botrel, Thomas Piellard, Youssef El Housni, Ivo Kubjas, and Arya Tabaie. 2022. *ConsenSys/gnark: v0.6.4*. https://doi.org/10.5281/zenodo.6093969

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

[10] Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. 2019. Data Poisoning Attacks to Local Differential Privacy Protocols. *CoRR* abs/1911.02046 (2019). arXiv:1911.02046 http://arxiv.org/abs/1911.02046

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

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

[13] Albert Cheu, Adam D. Smith, and Jonathan R. Ullman. 2021. Manipulation Attacks in Local Differential Privacy. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. IEEE, 883–900. https://doi.org/10.1109/SP40001.2021.00001

[14] Guillaume Drevon and Aleksander Kampa. 2019. Benchmarking Zero-Knowledge Proofs with isekai. https://sikoba.com/docs/SKOR_isekai_benchmarking_201912.pdf

[15] Cynthia Dwork. 2006. Differential Privacy. In *Automata, Languages and Programming*, Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–12.

[16] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography*, Shai Halevi and Tal Rabin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 265–284.

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

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

[19] Jens Groth. 2016. On the Size of Pairing-based Non-interactive Arguments. *IACR Cryptol. ePrint Arch.* (2016), 260. http://eprint.iacr.org/2016/260

[20] Zhangshuang Guan, Zhiguo Wan, Yang Yang, Yan Zhou, and Butian Huang. 2022. BlockMaze: An Efficient Privacy-Preserving Account-Model Blockchain Based on zk-SNARKs. *IEEE Transactions on Dependable and Secure Computing* 19, 3 (2022), 1446–1463. https://doi.org/10.1109/TDSC.2020.3025129

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

[22] Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. 2021. Preventing Manipulation Attack in Local Differential Privacy Using Verifiable Randomization Mechanism. In *Data and Applications Security and Privacy XXXV - 35th Annual IFIP WG 11.3 Conference, DBSec 2021, Calgary, Canada, July 19-20, 2021, Proceedings (Lecture Notes in Computer Science, Vol. 12840)*, Ken Barker and Kambiz Ghazinour (Eds.). Springer, 43–60. https://doi.org/10.1007/978-3-030-81242-3_3

[23] Gonzalo Munilla Garrido, Johannes Sedlmeir, and Matthias Babel. 2022. Towards Verifiable Differentially-Private Polling. In *Proceedings of the 17th International Conference on Availability, Reliability and Security*. 1–11.

[24] Satoshi Nakamoto. 2009. Bitcoin: A peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf

[25] Arjun Narayan, Ariel Feldman, Antonis Papadimitriou, and Andreas Haeberlen. 2015. Verifiable Differential Privacy. In *Proceedings of the Tenth European Conference on Computer Systems* (Bordeaux, France) *(EuroSys '15)*. Association for Computing Machinery, New York, NY, USA, Article 28, 14 pages. https://doi.org/10.1145/2741948.2741978

[26] Deloitte NextGen. 2022. Enterprise Blockchain. https://www2.deloitte.com/dk/da/pages/technology/nextgen-blockchain.html, accessed on 29 November 2022.

[27] Bank of Canada, European Central Bank, Bank of Japan, Sveriges Riksbank, Swiss National Bank, Bank of England, Board of Governors Federal Reserve System, and Bank for International Settlements. 2020. Central bank digital currencies: foundational principles and core features. https://www.bis.org/publ/othp33.pdf

[28] Torben Pryds Pedersen. 1992. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In *Advances in Cryptology — CRYPTO '91*, Joan Feigenbaum (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 129–140.

[29] Georgia Tsaloli and Aikaterini Mitrokotsa. 2019. Differential Privacy meets Verifiable Computation: Achieving Strong Privacy and Integrity Guarantees. In *Proceedings of the 16th International Joint Conference on e-Business and Telecommunications, ICETE 2019 - Volume 2: SECRYPT, Prague, Czech Republic, July 26-28, 2019*, Mohammad S. Obaidat and Pierangela Samarati (Eds.). SciTePress, 425–430. https://doi.org/10.5220/0007919404250430

[30] Nicolas Van Saberhagen. 2013. CryptoNote v 2.0. (2013). Retrieved from https://github.com/monero-project/research-lab/blob/master/whitepaper/whitepaper.pdf.

[31] Stanley L Warner. 1965. Randomized response: A survey technique for eliminating evasive answer bias. *J. Amer. Statist. Assoc.* 60, 309 (1965), 63–69.

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