Cryptographic Commitments on Anonymizable Data

Xavier Bultel
INSA Centre Val de Loire,
University of Orléans, Inria-Saclay
France
xavier.bultel@insa-cvl.fr

Céline Chevalier
Panthéon-Assas University,
ENS, PSL University, CNRS, Inria
France
celine.chevalier@ens.fr

Charlène Jojon
INSA Centre Val de Loire,
University of Orléans, Inria-Saclay
France
charlene.jojon@insa-cvl.fr

Diandian Liu
INSA Centre Val de Loire,
University of Orléans
France
diandian.liu@insa-cvl.fr

Benjamin Nguyen
INSA Centre Val de Loire,
University of Orléans, Inria-Saclay
France
benjamin.nguyen@insa-cvl.fr

***Abstract*—Local Differential Privacy (LDP) mechanisms consist of (locally) adding controlled noise to data in order to protect the privacy of their owner. In this paper, we introduce a new cryptographic primitive called LDP commitment. Usually, a commitment ensures that the committed value cannot be modified before it is revealed. In the case of an LDP commitment, however, the value is revealed after being perturbed by an LDP mechanism. Opening an LDP commitment therefore requires a proof that the mechanism has been correctly applied to the value, to ensure that the value is still usable for statistical purposes. We also present a security model for this primitive, in which we define the hiding and binding properties. Finally, we present a concrete scheme for an LDP staircase mechanism (generalizing the randomized response technique), based on classical cryptographic tools and standard assumptions. We provide an implementation in Rust that demonstrates its practical efficiency (the generation of a commitment requires just a few milliseconds).**

On the application side, we show how our primitive can be used to ensure simultaneously privacy, usability and traceability of medical data when it is used for statistical studies in an open science context. We consider a scenario where a hospital provides sensitive patients data signed by doctors to a research center after it has been anonymized, so that the research center can verify both the provenance of the data (*i.e.* verify the doctors' signatures even though the data has been noised) and that the data has been correctly anonymized (*i.e.* is usable even though it has been anonymized).

1. Introduction

Individual's medical data presents the specific characteristic of being both used (i) by practitionners to help diagnose and propose treatments to patients (primary use) and (ii) by researchers as input data for clinical studies or statistical studies (secondary use). When data is used for treatment purposes, it must obviously be as precise as possible. However, in the context of large scale statistical studies, this data may benefit from additional privacy protection, such as (local) differential privacy (LDP) [1], [2], which is a standard model that provides formal statistical guarantees on the privacy of the data shared (described formally in Section 2). Indeed, there are currently many works where the use of such privacy preserving techniques are applied before performing data analysis (see [3] for a survey), quite often in the context of federated learning [4].

Reproducible Medical Research. Our context is illustrated in Fig. 1, where numbers in brackets refer to the different steps shown in this figure. The application pertains to reproducibility in medical research (RR) [5], where data, produced by Diana, a medical practitioner who is Alice and Bob's doctor is primarily used by Helen the Hospital to treat both patients. Diana, Alice and Bob agree on a *secondary* use of the data: sending it to Robin, a researcher from a different institution, who is performing some study on Alice and Bob's illness. However, for confidentiality reasons, this data may be anonymized in particular if it will subsequently be published for RR purposes, which is often done using *LDP* [3]. After a discussion, Diana and Alice (1a), and Diana and Bob (1b) agree on the privacy parameters to apply to this secondary use of their medical data [6]. Note that Alice's data is automatically extracted by the medical devices, and its authenticity can also be automatically signed, while Diana will certify Bob's medical data. Both Alice and Bob's data, information on the anonymization process and parameters chosen are then sent to the datacenter of the hospital (managed by Helen) that Diana works for (2a-b). At a later point in time (potentially when neither Diana, Bob nor Alice are available), Robin, a researcher from a different institution, interested in performing a large scale medical study, contacts Helen in order to obtain the hospital data. Helen transmits this authenticated data with local differential privacy constraints to Robin (3). In order to comply with RR and legal principles, Robin must on the one hand publish the data he used to perform his study, letting reviewers use the *same data* to confirm his results, and on the other hand he must prove that his data was obtained through a correct and legitimate anonymization process. In our context, this means proving that the whole data collection and anonymization task was

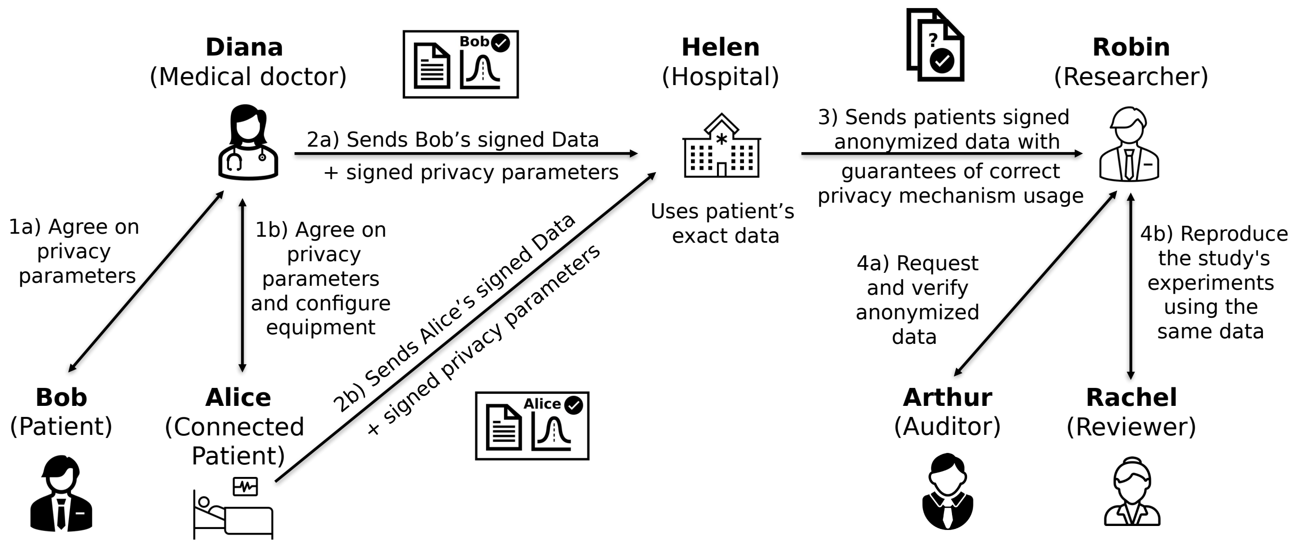
Diana
(Medical doctor)
Bob
Helen
(Hospital)
Robin
(Researcher)
2a) Sends Bob's signed Data
+ signed privacy parameters
3) Sends patients signed anonymized data with guarantees of correct privacy mechanism usage
Uses patient's exact data
1a) Agree on privacy parameters
1b) Agree on privacy parameters and configure equipment
2b) Sends Alice's signed Data
+ signed privacy parameters
4a) Request and verify anonymized data
4b) Reproduce the study's experiments using the same data
Bob
(Patient)
Alice
(Connected Patient)
Alice
Arthur
(Auditor)
Rachel
(Reviewer)

Figure 1. Medical research example

performed correctly by Helen on data authenticated by the data producer, i.e. Diana. Thus Helen must provide Robin with a proof that she has correctly applied the differential privacy mechanism to the data signed by Diana. Robin is thus able to convince both Arthur, an auditor for a data protection authority (4a), and Rachel, a reviewer trying to reproduce Robin's results (4b), that the certified data used in his experiments was produced and processed legitimately.

Threat Model. Therefore, we consider the following threat model, which is summarized in Fig. 2. First, the doctor (Diana) and her patients (whom we will all consider as a single entity under the name Diana) trust Helen (the hospital manager) because they agree to provide her with their exact (non-anonymized) data in order to ensure optimal treatment for the patients. Robin (the researcher) and Helen trust Diana to sign the patient's real data, assuming that it is not in the patient's interest to give false information to the hospital. It is also important to note that we consider that Helen and Robin **have no interest in colluding**, *e.g.* that Helen is not going to send (covertly) the non-anonymized data to Robin, while also providing Robin with proven anonymized data, as per the protocol. While this may be considered a strong hypothesis and a potential weakness in the threat model, it is justified by the context of RR, where Robin needs to be able to prove that the exact data he used was obtained legitimately. Indeed, in the case of a collusion, Arthur will discover that despite Robin being able to show he owns legitimate authenticated anonymized data, his results cannot be reproduced (*e.g.* if Robin used the non-anonymized data directly send by Helen). Arthur and Rachel are trusted by all the other parties, as they are independant authorities only performing reviewing and audit.

Selling private data. Another application that could directly benefit from the results of our work is the sale of private data by data brokers. The seminal article on this topic [7] considers a marketplace where users (Diana) sell their private data (with controlled added differentially private noise) and buyers (Robin) purchase this data from third party data brokers (Helen), where the price depends

	Trusts Diana	Trusts Helen	Trusts Robin	Trusts Arthur and Rachel
Diana	–	Yes	No	Yes
Helen	Yes	–	No	Yes
Robin	Yes	No	–	Yes
Arthur and Rachel	Yes	No	No	–

Figure 2. Threat model.

on the noise. In the context of the initial article, the third party must be trusted by both buyer and seller to add the correct amount of noise. Our approach allows the *removal* the trust hypothesis of the buyer on this third party. Indeed, our approach lets Helen prove to Robin the exact amount of noise that was added, ensuring a fair price.

1.1. Our contributions

On a technical level, the first contribution of this paper is the formalization and security modelling of a new primitive called *LDP Commitment*, which allows a user to commit a value in such a way that this value can be opened (potentially by another user) after the application of a local differential privacy mechanism. The verifier must be convinced that this mechanism has been applied correctly. Of course, only the anonymized data, and not the original data, is required to verify the correct opening of the commitment. Note that the LDP mechanism is inherently probabilistic and requires the use of a random seed. This seed cannot be fully chosen by the user committing the value and/or opening the commitment, as this would allow them to control the data after anonymization. Nor can the seed be fully chosen by the verifier, who could at best choose a seed that minimizes noise and at worst reverse the anonymization mechanism. It is therefore necessary that the choice of this seed should be shared between the data committer (at the time of commitment, it must not be possible for the opener to change it afterwards) and the verifier. Thus, if at least one of them chooses their nonce at random, the result is a perfectly random seed. In practice, in this setting the data owner has no interest whatsoever in not choosing a truly random nonce, since this guarantees the protection of their data.

First, we extend the classic hiding property for commitments (which guarantees that the committed data is kept private until the commitment is opened) to our primitive. We then define the LDP-hiding property, which ensures that the verifier obtains no more information about the committed value than the anonymized value. We give two formalisms for this property (real/simulation based and indistinguishability based): the first considers an adversary who must guess whether they received a real commitment or whether the commitment was simulated without using the value, and the second considers an adversary who chooses two values and must guess which of the two was committed and then opened. Unlike other indistinguishability games where the probability of success is compared to 1/2 (the probability of guessing the hidden value among two), indistinguishability based LDP-hiding requires specifying the probability of success of an adversary who would only exploit the anonymized message to find the original value. We give a simple formula to express this probability. We also show that classical hiding coupled with the real/simulation version implies the indistinguishability version.

We also extend the binding property (which ensures that a commitment can be open to unique value) to our primitive. We define the notion of LDP-binding, which catches an adversary trying not to apply the LDP mechanism correctly when they open the commitment. We model an adversary who commits a value and has to open it using a randomly generated seed. The scheme is considered to have the LDP-binding property if the distribution produced by this opening is the same as that produced by the application of the LDP mechanism on the value committed by the adversary.

Our second technical contribution is to propose a scheme for an LDP staircase mechanism (generalization of randomized response) and to prove its security in our model. Remarkably, our scheme uses only simple and efficient well-established standard cryptographic tools: it is based on the decisional Diffie–Hellman (DDH) assumption in a group of prime order and uses Schnorr-based zero-knowledge proofs. The complexity in terms of commitment size and computation time increases only logarithmically with the size of the set on which the data is chosen and the inverse of the probability of obtaining the real answer in the staircase. This makes the scheme very efficient in practice, as we show by analyzing the performance of a Rust implementation. In particular, the opening check is very efficient because it requires a constant number of exponentiations in the group, which is the most expensive operation of our algorithms by a huge margin. The commitment correction check takes a little longer but can be pre-computed before opening. In a nutshell, all our algorithms take a few milliseconds for standard parameters (when considering sets of a hundred responses) and less than 30 milliseconds in extreme cases (when considering sets of more than a billion responses) on a regular personal computer.

This new primitive allows us to address our problem: doctor Diana can sign the description of some medical value together with the commitment of this value (for example, ”Self-assessment of pain, $\mathsf{Commit}(7)$”, where Commit is an LDP commitment algorithm). Note that the commitment, which is signed, gives no information about the committed value (in our example 7). Moreover, this value is not needed to verify the signature. The value, the description, the signature, the commitment, and corresponding opening key are sent to Helen, the hospital manager, who stores them. If Helen wishes to transmit this value after applying the LDP mechanism, she sends the signature to a verifier with the opening of the commitment on the anonymized value. By verifying that the commitment has been opened correctly, verifier Robin is convinced that the LDP mechanism has indeed been applied to the anonymized value transmitted by Helen, and by verifying the signature on the commitment, Robin is convinced that the original committed value (before anonymization) does indeed come from Diana. However, Robin learns nothing about this original value. Thus, Helen cannot cheat on the application of the LDP mechanism to the value signed by Diana. Furthermore, in the event of an audit by Arthur due to the leakage of a patient's personal data, Arthur can consult the proof of opening of the commitment and the signature on the commitment to ensure that the data sent by Helen is correctly anonymized. Note that our implementation efficiency analysis takes into account the signature of the commitment.

1.2. Related work

In 2008, Ambainis, Jakobsson and Lipmaa first proposed the use of cryptographic tools to secure LDP mechanisms in [8]. Their work focuses on a randomized response mechanism where the response is a bit and the probability of giving that response depends on a parameter. They propose interactive protocols between a responder and a verifier where the verifier can check that the responder has correctly applied the LDP mechanisn on their response, without learning any additional information about the exact response. Our work can be seen as an extension of this first work, where the response set is not limited to one bit and where the data can be committed before being revealed, potentially by an untrusted delegate. Note also that our protocol does not require any interaction between the parties.

In recent years, the growing interest in differencial privacy and recent studies showing its critical weakness in the front of manipulation attacks [9], where respondents attempt to bias the results of statistics by manipulating their answers, have led to renewed interest in this approach. In [10], Kato *et. al.* extends the work of Ambainis *et. al.* [8] to other randomization mechanisms. Note that as in [8], their approach is interactive, and considers a different scenario where the data owners (in our case, the doctors) interact directly with the verifier, so it cannot be applied directly to our use case.

In [11], Biswas and Cormode extend the concept of verifiable differential privacy to the non-local case, where multiple entities commit their data, and a server computes a function on that data, reveals the anonymized result, and proves that it has correctly applied the differential privacy mechanism. Their approach consists of committing data with a partially homomorphic commitment, which makes it possible to apply the counting query directly to the committed values, then to demonstrate with zero-knowledge proofs that the result was opened by correctly applying the binomial DP mechanism [1]. The proofs used are interactive, but it may be possible to make them non-interactive without significant modification. By directly entering an integer value and signing the commitment, it might be possible to adapt this work for our use case, considering another type of values (integers in an infinite sets instead of integers in an interval) and another LDP mechanism (the binomial mechanism instead of the generalised randomised response). Although it is difficult to compare this work because of the use of different DP mechanisms, we show through our implementation that our solution is more efficient for similar privacy parameters (only a few milliseconds vs. around a minute). Moreover, we propose a more general and complete security model than in [11] that can be applied to solutions that are not unique zero-knowledge proof protocols.

In [12], the authors use generic zero-knowledge proofs for arithmetic circuits (SNARKs) to verify that a data item has been correctly anonymized using a local differential privacy mechanism. They use Circome [13] to implement the circuit corresponding to the randomized response mechanism (non-generalized, on a single bit) and the exponential mechanism on a set of 128 values. They use the Groth-16 [14] SNARKs to experiment with their construction. The authors mention that their proof could be used on data signed in an anonymous credential, resulting in a scheme similar to ours, however, this would require the use of a much more complex circuit than the one actually presented in their paper, making it possible to prove knowledge of a valid (hidden) signature on the (hidden) data used by the DP mechanism. SNARKs are generic and easy to instantiate, with the advantage of providing a commitment and proof of a (small) constant size. However, they use fewer standard tools (*e.g.* pairings) and weak cryptographic security assumptions (*e.g.* the generic group model), require a rather large setup, and more expensive computation and verification time (for Groth-16 [14] the runtime depends on the complexity of the circuit used, *i.e.* the number of exponentiations required for the proof grows linearly with the number of wires and multiplication gates in the proven arithmetic circuit). Our proposal offers a more efficient alternative: even without the signature in the circuit, and despite providing less privacy and using a smaller set of values, proof generation takes a few seconds for SNARKs compared to only a few milliseconds for our prototype. Note that the size of the commitments and the proof grows logarithmically with our parameters, but still seems acceptable (a few kilobytes for the largest parameters). Moreover, [12] does not model and prove the security of their construction. We believe that our model could be used for this purpose.

In [15], the authors consider a setup where users encrypt their data with keys provided by a Cryptographic Service Provider (CSP), and send them to a Data Analyst (DA). As the encryption is homomorphic, the DA can query on it, and ask the CSP to decrypt the result by adding noise with a DP mechanism. If the general configuration is similar to ours (the users, CSP and DA are respectively Diana, Helen and Robin), their aim differs substantially from ours since they do not provide any mechanism for verifying that the CSP correctly adds noise to the results, which is the core of our work.

Another line of work consists in using multiparty computation [16] or homomorphic encryption [17] to compute statistical functions from the private data of several users with differential privacy, either in the presence of honest-but-curious adversaries [18], or malicious adversaries [16]. The purpose of these works differs significantly from ours, since it considers the participation of several interacting data owners to compute an overall result perturbed via DP in a secure way, so that there is no question of working on values committed and signed beforehand by a user who is offline at the time the data is perturbed, nor of keeping proof that the DP mechanism was correctly applied.

Finally, several signature primitives allow delegating the power to modify/sign messages. Sanitizable signatures [19], [20] allow a delegate to modify certain parts of a signed message, but the restrictions on these modifications are either too specific [21] or too weak [19] to be used for differential privacy mechanisms. Functional signatures [22] and delegatable functional signatures [23] allow a user to sign messages that verify certain functions or predicates, policy-based signatures [24] allow a user to sign a message if it satisfies a certain policy, and homomorphic signatures [25] allow computations to be performed on signed data. These primitives could be considered for our problem, but their genericity implies a significant loss of efficiency (due to the use of heavy tools, such as proofs for circuits), whereas the aim of this work is to provide a simple and optimized solution, tailored for local differential privacy.

2. Background

In this section, we introduce our notations as well as the tools that we need, both from cryptography and (local) differential privacy.

We denote by $[\![n]\!]$ the set $\{1, \cdots, n\}$ where n is an integer. By $x \leftarrow y$ we mean that the variable x takes a value y, by $x \leftarrow \mathsf{Algo}(y)$ that the variable x takes a value outputted by algorithm Algo on input y, and by $r \overset{\$}{\leftarrow} S$ that r is chosen from the uniform distribution on S. Finally $r_1, \cdots, r_n \overset{\$}{\leftarrow} S$ means that $\forall i \in [\![n]\!], r_i \overset{\$}{\leftarrow} S$. We use the acronym PPT for *probabilistic polynomial time*, s.t. for *such that* and λ denotes the security parameter.

Cryptographic tools. We recall the Decisional Diffie-Hellman (DDH) assumptions in a group $\mathbb{G} = \langle g \rangle$ of prime order p. The DDH assumption states that given a random $(x, y, z) \in (\mathbb{Z}_p^*)^3$, it is difficult for a PPT algorithm $\mathcal{D}$ to distinguish between $(g^x, g^y, g^{x \cdot y})$ and (g^x, g^y, g^z).

Definition 1 (Decisional Diffie-Hellman (DDH) assumption). *Let $\mathbb{G} = \langle g \rangle$ be a multiplicative group of prime order p. The* decisional Diffie-Hellman *(DDH) assumption states that there exists a negligible function ϵ_{DDH} s.t. for any* PPT *algorithm $\mathcal{D}$,*

$$|\Pr[(x, y) \overset{\$}{\leftarrow} (\mathbb{Z}_p^*)^2 \ : \ 1 \leftarrow \mathcal{D}(g^x, g^y, g^{x \cdot y})] - \Pr[(x, y, z) \overset{\$}{\leftarrow} (\mathbb{Z}_p^*)^3 \ : \ 1 \leftarrow \mathcal{D}(g^x, g^y, g^z)]| \leq \epsilon_{\mathsf{DDH}}(\lambda).$$

We recall the concept of Non-Interactive Zero Knowledge Proofs (NIZKP), that allows a prover to prove the knowledge of a witness matching a statement to a verifier.

Definition 2 (Non-Interactive Proofs (NIP) [26]). *Let $\mathcal{R}$ be a binary relation and let $\mathcal{L}$ be a language s.t. $s \in \mathcal{L} \Leftrightarrow (\exists w, (s, w) \in \mathcal{R})$. A* Non-Interactive Proof *(NIP) for $\mathcal{L}$ is a couple of algorithms* $(\mathsf{NIP}, \mathsf{NIPVerify})$ *s.t.:*
$\underline{\mathsf{NIP}\{w : (s, w) \in \mathcal{R}\}}$**.** *This algorithm outputs a proof π.*
$\underline{\mathsf{NIPVerify}(s, \pi)}$**.** *This algorithm outputs a bit b.*

A Non-Interactive Zero Knowledge Proof (NIZKP) is a NIP having the following properties:
Correctness. *For any $s, w, \mathcal{R}$ s.t. $(s, w) \in \mathcal{R}$ and $\pi \leftarrow$* $\mathsf{NIP}\{w : (s, w) \in \mathcal{R}\}$, $\mathsf{NIPVerify}(s, \pi)$ *returns 1.*
Extractability. *There exists a PPT knowledge extractor Ext and a negligible function ϵ_{ext} s.t. for any algorithm $\mathcal{A}^{\mathsf{Sim}(\cdot)}(\lambda)$ having access to a simulator that forges signatures for chosen statement and that outputs a fresh pair (s, π) with* $\mathsf{NIPVerify}(s, \pi) = 1$*, the extractor $\mathsf{Ext}^{\mathcal{A}}(\lambda)$ outputs w s.t. $(s, w) \in \mathcal{R}$ having access to $\mathcal{A}(\lambda)$ with probability at least $1 - \epsilon(\lambda)$.*
Zero-knowledge. *A proof π* leaks no information, *i.e., there exists a polynomial time algorithm Sim (called the* simulator*) s.t.* $\mathsf{NIP}\{w : (s, w) \in \mathcal{R}\}$ *and $\mathsf{Sim}(s)$ follow the same probability distribution.*

Local Differential Privacy (LDP). Local differential privacy [2] (LDP) is a well accepted standard definition to measure the security (privacy) of randomized algorithms in the privacy domain. Intuitively, local differential privacy is a constraint on the distribution of the outcomes of the algorithm, when an attacker observes the output of a single execution of the algorithm. It is well adapted in the context of independant devices or individuals producing results that must present formal privacy guarantees. LDP can be seen as a non-centralized version of the differential privacy model [1]. Formally,

Definition 3 (Local Differential Privacy [2]). *A randomized algorithm $\mathcal{A}$ satisfies ϵ-local differential privacy (LDP) where $\epsilon \in \mathbb{R}^{+*}$, if for any pair of input values $(x_1, x_2) \in Domain(\mathcal{A})$ and any possible output $y \in Codomain(\mathcal{A})$:*

$$\Pr[\mathcal{A}(x_1) = y] \leq \exp(\epsilon) \cdot \Pr[\mathcal{A}(x_2) = y].$$

Note that the greater the value of ϵ (called *privacy budget* in differential privacy models), the lesser the privacy of the mechanism. Another important characteristic of LDP is the *utility* of the mechanism, which is often measured by the difference between two distributions (called f-divergence) [27]. Indeed, the authors of [27] have shown that there exists a class of optimal mechanisms (*i.e.* which maximize utility for a given ϵ value) called *staircase mechanisms*. These mechanisms satisfy the following constraints:

- the output domain size (noted $|\mathcal{Y}|$) is at most the input domain size (noted $|\mathcal{X}|$); and
- for all $y \in \mathcal{Y}$, and $x, x' \in \mathcal{X}$, $\left|\ln \frac{\Pr(\mathcal{A}(x=y))}{\Pr(\mathcal{A}(x'=y))}\right| \in \{0, \epsilon\}$ with $\epsilon > 0$.

This means that an optimal mechanism $\mathcal{A}$ can only output values with two different probabilities p_1 and p_2 with $p_1 \geq p_2$ s.t. $p_1 = \exp(\epsilon) \cdot p_2$. Thus, for a mechanism with a bounded input domain and output domain as in our case, the simple optimal mechanism that we consider is the following (noted LDP in the rest of the article):

Definition 4 (Optimal mechanism LDP). *For all $x \in \mathcal{X}$, $y \in \mathcal{Y}$, and for a given $\epsilon \in \mathbb{R}^{+*}$,*

- $\mathsf{LDP}(x)$ *outputs y with probability p_1 if $x = y$*
- $\mathsf{LDP}(x)$ *outputs y with probability p_2 if $x \neq y$*
- $p_1 = \exp(\epsilon) \cdot p_2$

We show in Section 4.1 how to compute probabilities p_1 and p_2 given the size of the domain $|\mathcal{X}|$.

3. Formal Model for LDP Commitment

In this section we give a formal definition and security model for LDP commitments.

3.1. Formal Definition

As with standard cryptographic commitments, an LDP commitment allows a user to commit a secret value in such a way that this value can be opened (*i.e.* revealed) using a key. The user consulting the opened value can use an algorithm to check that the commitment has been correctly opened, *i.e.* that the opened value is indeed the one that was committed. In addition, an LDP commitment also depends on a public LDP mechanism. A user with the key can, from a random seed, use an alternative opening algorithm that reveals the anonymized value (to which the LDP mechanism was applied). In this case, the user also produces a proof that they have correctly applied the opening algorithm, without revealing anything more about the actual committed value. Finally, the commitment algorithm also produces a proof that the commitment was indeed generated by the LDP mechanism, ensuring that the released value is indeed the value that was committed to by applying the given LDP mechanism. This random seed must be chosen by someone other than the user opening the commitment (either by a trusted third party or directly by the verifier), otherwise the user could choose the seed that reveals the value they chose instead of following the LDP mechanism distribution.

Definition 5 (LDP Commitment). *A* LDP Commitment (LDP-C) *is a tuple of polynomial time algorithms* $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{OpenLDP}, \mathsf{VerOpen}, \mathsf{VerOpenLDP}, \mathsf{VerCommit})$ *s.t.:*
$\underline{\mathsf{Setup}(\lambda, \mathsf{LDP})}$**:** *takes as input the security parameter λ, and a LDP mechanism* LDP*; returns a setup* set *containing a message set $\mathcal{M}$ and a seed set Θ.*
$\underline{\mathsf{Commit}(m, \theta)}$**:** *takes as input a message m and a random seed θ; returns a secret key k, a commitment c and a proof π_*.*
$\underline{\mathsf{VerCommit}(\mathsf{c}, \pi_*)}$**:** *takes as input a commitment c and a proof π_*; returns an acceptance bit b.*
$\underline{\mathsf{Open}(\mathsf{k}, \mathsf{c})}$**:** *takes as input a secret key k and a commitment c; returns a message m and a proof π.*
$\underline{\mathsf{VerOpen}(\mathsf{c}, m, \pi)}$**:** *takes as input a commitment c, a message m, and a proof π; returns an acceptance bit b.*
$\underline{\mathsf{OpenLDP}(\mathsf{k}, \mathsf{c}, \hat{\theta})}$**:** *takes as input a secret key k, a commitment c, and a random seed $\hat{\theta}$; returns a message $\hat{m}$ and a proof $\hat{\pi}$.*
$\underline{\mathsf{VerOpenLDP}(\mathsf{c}, \hat{m}, \hat{\pi}, \hat{\theta})}$**:** *takes as input a commitment c, a message $\hat{m}$, a proof $\hat{\pi}$, and a seed $\hat{\theta}$; returns an acceptance bit b.*

We next define correctness. Our definition requires that honest openings made on honestly generated commitments will always be verified, and that the commitment opening using the LDP mechanism will follow the correct probabilistic distribution if the committer (resp. the verifier) is honest (*i.e.* if they choose their θ seed truly randomly).

Definition 6 (Correctness). *A* $\mathsf{LDP\text{-}C}$ *is said to be correct for* LDP *it respects the two following conditions:*

1) *For any* $\lambda \in \mathbb{N}$*, any* set *generated from* $\mathsf{Setup}(\lambda, \mathsf{LDP})$ *and containing the sets* $\mathcal{M}$ *and* Θ*, any* $m \in \mathcal{M}$*, any* $\theta \in \Theta$*, any* $\hat{\theta} \in \Theta$*, any* $(\mathsf{k}, \mathsf{c}, \pi_*)$ *generated from* $\mathsf{Commit}(m, \theta)$*, any* (m', π) *generated from* $\mathsf{Open}(\mathsf{k}, \mathsf{c})$*, and any* $(\hat{m}, \hat{\pi})$ *generated from* $\mathsf{OpenLDP}(\mathsf{k}, \mathsf{c}, \hat{\theta})$*, it holds that* $\mathsf{VerCommit}(\mathsf{c}, \pi_*) = \mathsf{VerOpen}(\mathsf{c}, m', \pi) = \mathsf{VerOpenLDP}(\mathsf{c}, \hat{m}, \hat{\pi}, \hat{\theta}) = 1$ *and* $m = m'$*.*
2) *For any* $\lambda \in \mathbb{N}$*, any* set *generated from* $\mathsf{Setup}(\lambda, \mathsf{LDP})$ *and containing the sets* $\mathcal{M}$ *and* Θ*, any* $(m, \hat{m}_0) \in \mathcal{M}^2$*, and any* $(\theta_0, \hat{\theta}_0) \in \Theta^2$*, it holds that:*

$$
\begin{aligned}
&\Pr\left[\hat{m}_1 \leftarrow \mathsf{LDP}(m) : \hat{m}_0 = \hat{m}_1\right] \\
&= \Pr\left[\begin{array}{ll} \theta_1 \xleftarrow{\$} \Theta; & \\ (\mathsf{k}, \mathsf{c}, \pi_*) \leftarrow \mathsf{Commit}(m, \theta_1); & : \hat{m}_0 = \hat{m}_1 \\ (\hat{m}_1, \hat{\pi}) \leftarrow \mathsf{OpenLDP}(\mathsf{k}, \mathsf{c}, \hat{\theta}_0) & \end{array}\right] \\
&= \Pr\left[\begin{array}{ll} \hat{\theta}_1 \xleftarrow{\$} \Theta; & \\ (\mathsf{k}, \mathsf{c}, \pi_*) \leftarrow \mathsf{Commit}(m, \theta_0); & : \hat{m}_0 = \hat{m}_1 \\ (\hat{m}_1, \hat{\pi}) \leftarrow \mathsf{OpenLDP}(\mathsf{k}, \mathsf{c}, \hat{\theta}_1) & \end{array}\right].
\end{aligned}
$$

3.2. Security Model

The natural properties of cryptographic commitments are *hiding*, which guarantees that the commitment does not reveal any information about the committed value before being opened, and *binding*, which guarantees that the commitment can only be opened in one way (by revealing the actual committed value). In our case, since there are several ways to open LDP commitments (with or without the LDP mechanism), we need to adapt these properties accordingly. Fig. 3 shows the experiments corresponding to the different variations of these properties we define.

3.2.1. Hiding. First of all, we define Hiding in the classical way: an adversary who chooses two messages and receives a commitment for one of the two messages is unable to distinguish which message is being used (its probability of distinguishing is close to $1/2$).

Definition 7 (Hiding). *Let* λ *be a security parameter,* LDP *be a LDP mechanism, and* P *be a* $\mathsf{LDP\text{-}C}$*.* P *is said to be* hiding *if for any* PPT *two-party algorithm* $\mathcal{A}$*, there exists a negligible function* ϵ *s.t.:*

$$
\left|\Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{Hiding}}_{P,\mathsf{LDP},\mathcal{A},0}(\lambda)\right] - \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{Hiding}}_{P,\mathsf{LDP},\mathcal{A},1}(\lambda)\right]\right| \leq \epsilon(\lambda),
$$

where the Hiding *experiment is given in Fig. 3.*

Note: As we use the standard notations from the security and differential privacy worlds, ϵ and $\epsilon(\lambda)$ represent different concepts. However, as we feel that their meaning is obvious from the context, we have kept these notations.

We then consider an adversary for whom we open a commitment with the algorithm $\mathsf{OpenLDP}$. In this case, the adversary is not expected to learn any more information about the original value than they learn from the anonymized value returned by the commitment opening algorithm $\mathsf{OpenLDP}$. To model this, we first propose an experiment ($\mathsf{ROS\text{-}LDP\text{-}Hiding}$) where the adversary has to guess whether they have received the biased value returned by $\mathsf{OpenLDP}$ (*real case* with $b = 0$), or whether they have received the biased value by the true LDP mechanism (*simulated case* with $b = 1$, in which case the proof of correction of the opening must be simulatable to the adversary). To do this, the adversary can choose the value committed, and can also choose the random seed used for the opening.

Definition 8 (Real Or Simulated (ROS) LDP-Hiding). *Let* λ *be a security parameter,* LDP *be a LDP mechanism, and* P *be a* $\mathsf{LDP\text{-}C}$*.* P *is said to be* $\mathsf{ROS\text{-}LDP\text{-}hiding}$ *if for any three-party* PPT *algorithm* $\mathcal{A}$*, there exists a negligible function* ϵ *and a* PPT *simulator* Sim *that takes as input a message* m *and a commitment* c*, and that returns a simulated proof* $\hat{\pi}$ *s.t.:*

$$
\begin{aligned}
&\Big|\Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{ROS\text{-}LDP\text{-}Hiding}}_{P,\mathsf{LDP},\mathcal{A},0}(\lambda)\right] - \\
&\Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{ROS\text{-}LDP\text{-}Hiding}}_{P,\mathsf{LDP},\mathcal{A},1}(\lambda)\right]\Big| \leq \epsilon(\lambda),
\end{aligned}
$$

(the $\mathsf{ROS\text{-}LDP\text{-}Hiding}$ *experiment is given in Fig. 3).*

Another way of formalizing this property, which is closer to standard Hiding, is to commit a message drawn from two messages chosen by the adversary, open it with $\mathsf{OpenLDP}$, and verify whether the adversary can distinguish which message has been committed. We define the corresponding security experiment as $\mathsf{IND\text{-}LDP\text{-}Hiding}$. However, the opened message, although anonymized, can leak significant information about the actual committed message. This security property cannot therefore be defined as the fact that the difference between the probability that the adversary answers correctly and the probability that it is wrong is negligible, as is usually the case in indistinguishability experiments. For the same reasons, it is not possible to compare the probability of the adversary distinguishing the message to $1/2$. We therefore need to evaluate the probability that an adversary, acting optimally, would have of recovering the original message from a message anonymized by the given LDP mechanism only (without knowing the commitment). To do this, we define the algorithm $\mathsf{OptiGuess}$, which takes as input two original messages (m_0, m_1) and an anonymized message $\hat{m}$ computed by applying the LDP mechanism to one of these messages, then returns b when m_b is the most likely original message knowing only the anonymized message $\hat{m}$. From a technical point of view, the $\mathsf{IND\text{-}LDP\text{-}Hiding}$ experiment is parameterized by a bit b. When $b = 0$, we observe the probability of the adversary correctly distinguishing the original message, otherwise, when $b = 1$, we observe the optimal probability (given by $\mathsf{OptiGuess}$) of finding the original message among the messages chosen by the adversary using only the anonymized value. The IND-LDP-hiding security is satisfied when the probability of success of $\mathcal{A}$ when $b = 0$ is significantly higher than in the case $b = 1$.

Definition 9 (Indistinguishability (IND) LDP-Hiding)**.** *Let λ be a security parameter,* LDP *be a LDP mechanism, and P be a* $\mathsf{LDP\text{-}C}$. *P is said to be* IND-LDP-hiding *if for any three-party* PPT *algorithm $\mathcal{A}$ s.t. if the following value is positive:*

$$\epsilon(\lambda) = \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{IND\text{-}LDP\text{-}Hiding}}_{P,\mathsf{LDP},\mathcal{A},0}(\lambda)\right] - \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{IND\text{-}LDP\text{-}Hiding}}_{P,\mathsf{LDP},\mathcal{A},1}(\lambda)\right]$$

then $\epsilon(\lambda)$ is negligible, where the $\mathsf{IND\text{-}LDP\text{-}Hiding}$ *experiment is given in Fig. 3.*

While this definition provides an intuitive way to understand the IND-LDP-hiding, it does not provide a precise way to evaluate $\Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{IND\text{-}LDP\text{-}Hiding}}_{P,\mathsf{LDP},\mathcal{A},1}(\lambda)\right]$. To overcome this issue, we evaluate the winning probability of the algorithm $\mathsf{OptiGuess}$ in Theorem 1 (proven in Appendix A) in a more precise and usable way. Interestingly, it depends only on the distribution induced by LDP. Since the adversary's response is replaced by $\mathsf{OptiGuess}$ in $\mathsf{Exp}^{\mathsf{IND\text{-}LDP\text{-}Hiding}}_{P,\mathsf{LDP},\mathcal{A},0}(\lambda)$, the probability that the experiment returns 1 is the winning probability of $\mathsf{OptiGuess}$ (Fig. 3), weighted by the probabilities of the adversary's choices for the (m_0, m_1).

Theorem 1. *For any tuple $(\lambda, \mathsf{LDP}, m_0, m_1)$ where the set of messages of the* LDP *mechanism is denoted by $\mathcal{M}$ and $(m_0, m_1) \in \mathcal{M}^2$, we have:*

$$\Pr\left[\begin{array}{l} b \xleftarrow{\$} \{0,1\}; \hat{m} \leftarrow \mathsf{LDP}(m_b); \\ b_* \leftarrow \mathsf{OptiGuess}(\lambda, \mathsf{LDP}, \hat{m}, m_0, m_1) \end{array} : b = b_*\right]$$
$$= \frac{1}{2} \sum_{\hat{m} \in \mathcal{M}} \max\left(\begin{array}{c} \Pr[\hat{m} \leftarrow \mathsf{LDP}(m_0)], \\ \Pr[\hat{m} \leftarrow \mathsf{LDP}(m_1)] \end{array}\right).$$

The following theorem shows a relation between our hiding properties.

Theorem 2. *A* $\mathsf{LDP\text{-}C}$ *that is both hiding and ROS-LDP-hiding is IND-LDP-hiding.*

Intuitively, the ROS-LDP-hiding property ensures that observing an anonymized message from the real LDP mechanism with a simulated proof is similar, from the adversary's point of view, to observing a message and a proof returned by the algorithm $\mathsf{OpenLDP}$. The only additional information on $m_{b'}$ that the adversary knows is therefore the commitment c, however the hiding property ensures that c gives no information on $m_{b'}$ that can be significantly exploited by a PPT adversary. Finally, the best strategy left to the adversary is to try to guess $m_{b'}$ using only the anonymized message $\hat{m}$, as in the case of $b = 1$.

In the following, we provide a sketch of this proof, giving the sequence of games [28], but omitting the reductions between the games. The full proof is available in Appendix I.

Proof sketch. We use the following sequence of games:
Game G_0: This game is the same as the $\mathsf{IND\text{-}LDP\text{-}Hiding}$ experiment in the case where $b = 0$.
Game G_1: This game is the same as the game G_0 except that the challenger replaces the zero-knowledge proof $\hat{\pi}$ by a simulated proof, we have $\Pr[G_0 \text{ returns } 1] = \Pr[G_1 \text{ returns } 1]$.
Game G_2: This game is the same as the game G_1 except that $\hat{m}_0$ is obtained by computing $\mathsf{LDP}(m_{b'})$. By reduction we can show that:

$$|\Pr[G_1 \text{ returns } 1] - \Pr[G_2 \text{ returns } 1]| \leq \epsilon_{\mathsf{ROS\text{-}LDP\text{-}Hiding}}(\lambda),$$

where $\epsilon_{\mathsf{ROS\text{-}LDP\text{-}Hiding}}$ is the advantage on the $\mathsf{ROS\text{-}LDP\text{-}Hiding}$ experiment.
Game G_3: This game is the same as the game G_2 except that $(\mathsf{k}, \mathsf{c}, \pi_*)$ is obtained by picking a random $m \xleftarrow{\$} \mathcal{M}$ and by computing $(\mathsf{k}, \mathsf{c}, \pi_*) \leftarrow \mathsf{Commit}(m, \theta)$. By reduction, we can show that:

$$|\Pr[G_2 \text{ returns } 1] - \Pr[G_3 \text{ returns } 1]| \leq \epsilon_{\mathsf{Hiding}}(\lambda),$$

where $\epsilon_{\mathsf{Hiding}}$ is the advantage on the Hiding experiment. In G_3, the only information the adversary knows about $m_{b'}$ is $\mathsf{LDP}(m_{b'})$, so their best strategy for guessing b' is to apply the algorithm $\mathsf{OptiGuess}(\lambda, \mathsf{LDP}, \hat{m}_1, m_0, m_1)$, and in this case, the probability that G_3 returns 1 is the same as the probability that the $\mathsf{IND\text{-}LDP\text{-}Hiding}$ experiment returns 1 in the case $b = 1$. It follows that the adversary's advantage $\epsilon(\lambda)$ over the experiment $\mathsf{IND\text{-}LDP\text{-}Hiding}$ is bounded by $\epsilon_{\mathsf{ROS\text{-}LDP\text{-}Hiding}}(\lambda) + \epsilon_{\mathsf{Hiding}}(\lambda)$. □

The IND-LDP-hiding property better fits the security concept we want to formalize, however, the ROS-LDP-hiding property appears to be easier to prove and simplifies the proofs for the IND-LDP-hiding property. It is also stronger under the hypothesis that the scheme is hiding.

3.2.2. Binding. First of all, we define $\mathsf{Binding}$ in the classical way: an adversary is unable to open a commitment in two different ways (revealing two different messages) using the algorithm Open.

Definition 10 (Binding)**.** *Let λ be a security parameter,* LDP *be a LDP mechanism, and P be a* $\mathsf{LDP\text{-}C}$. *P is said to be* binding *if for any* PPT *algorithm $\mathcal{A}$, there exists a negligible function ϵ s.t.:*

$$\Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{Binding}}_{P,\mathsf{LDP},\mathcal{A}}(\lambda)\right] \leq \epsilon(\lambda),$$

where the $\mathsf{Binding}$ *experiment is given in Fig. 3.*

In the same way, we define the $\mathsf{LDP\text{-}Binding}$ experiment for the $\mathsf{OpenLDP}$ algorithm. LDP-Binding security is achieved when no PPT adversary is able to open a commitment on two different messages using the same seed on the algorithm $\mathsf{OpenLDP}$.

Definition 11 (LDP-Binding)**.** *Let λ be a security parameter,* LDP *be a LDP mechanism, and P be a* $\mathsf{LDP\text{-}C}$. *P is said to be* $\mathsf{LDP\text{-}binding}$ *if for any* PPT *algorithm $\mathcal{A}$, there exists a negligible function ϵ s.t.:*

$$\Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{LDP\text{-}Binding}}_{P,\mathsf{LDP},\mathcal{A}}(\lambda)\right] \leq \epsilon(\lambda),$$

where the $\mathsf{LDP\text{-}Binding}$ *experiment is given in Fig. 3.*

When the seed for algorithm $\mathsf{OpenLDP}$ is chosen randomly, the message opened should follow the same distribution as if it had been generated honestly using the

```
$\mathsf{Exp}^{\mathsf{Hiding}}_{P,\mathsf{LDP},\mathcal{A},b}(\lambda)$:
$\mathsf{set} \leftarrow \mathsf{Setup}(\lambda, \mathsf{LDP})$
$(m_0, m_1) \leftarrow \mathcal{A}_1(\mathsf{set})$
$\theta \xleftarrow{\$} \Theta$
$(\mathsf{k}, \mathsf{c}, \pi_*) \leftarrow \mathsf{Commit}(m_b, \theta)$
$b_* \leftarrow \mathcal{A}_2(\mathsf{c}, \pi_*)$
return $b_* = b$

$\mathsf{Exp}^{\mathsf{ROS\text{-}LDP\text{-}Hiding}}_{P,\mathsf{LDP},\mathcal{A},b}(\lambda)$:
$\mathsf{set} \leftarrow \mathsf{Setup}(\lambda, \mathsf{LDP})$
$m \leftarrow \mathcal{A}_1(\mathsf{set})$
$\theta \xleftarrow{\$} \Theta$
$(\mathsf{k}, \mathsf{c}, \pi_*) \leftarrow \mathsf{Commit}(m, \theta)$
$\hat{\theta} \leftarrow \mathcal{A}_2(\mathsf{c}, \pi_*)$
$(\hat{m}_0, \hat{\pi}_0) \leftarrow \mathsf{OpenLDP}(\mathsf{k}, \mathsf{c}, \hat{\theta})$
$\hat{m}_1 \leftarrow \mathsf{LDP}(m)$
$\hat{\pi}_1 \leftarrow \mathsf{Sim}(\hat{m}_1, \mathsf{c})$
$b_* \leftarrow \mathcal{A}_3(\hat{m}_b, \hat{\pi}_b)$
return $b_* = b$

$\mathsf{Exp}^{\mathsf{IND\text{-}LDP\text{-}Hiding}}_{P,\mathsf{LDP},\mathcal{A},b}(\lambda)$:
$\mathsf{set} \leftarrow \mathsf{Setup}(\lambda, \mathsf{LDP})$
$(m_0, m_1) \leftarrow \mathcal{A}_1(\mathsf{set})$
$b' \xleftarrow{\$} \{0, 1\}$
$\theta \xleftarrow{\$} \Theta$
$(\mathsf{k}, \mathsf{c}, \pi_*) \leftarrow \mathsf{Commit}(m_{b'}, \theta)$
$\hat{\theta} \leftarrow \mathcal{A}_2(\mathsf{c}, \pi_*)$
$(\hat{m}_0, \hat{\pi}) \leftarrow \mathsf{OpenLDP}(\mathsf{k}, \mathsf{c}, \hat{\theta})$
$\hat{m}_1 \leftarrow \mathsf{LDP}(m_{b'})$
if $b = 0$, then $b_* \leftarrow \mathcal{A}_3(\hat{m}_0, \hat{\pi})$
else $b_* \leftarrow \mathsf{OptiGuess}(\lambda, \mathsf{LDP}, \hat{m}_1, m_0, m_1)$
return $b_* = b'$

$\mathsf{Exp}^{\mathsf{Binding}}_{P,\mathsf{LDP},\mathcal{A}}(\lambda)$:
$\mathsf{set} \leftarrow \mathsf{Setup}(\lambda, \mathsf{LDP})$
$(m_0, m_1, \pi_0, \pi_1, \mathsf{c}) \leftarrow \mathcal{A}(\mathsf{set})$
return $(m_0 \neq m_1 \wedge \mathsf{VerOpen}(\mathsf{c}, m_0, \pi_0) \wedge \mathsf{VerOpen}(\mathsf{c}, m_1, \pi_1))$

$\mathsf{Exp}^{\mathsf{LDP\text{-}Binding}}_{P,\mathsf{LDP},\mathcal{A}}(\lambda)$:
$\mathsf{set} \leftarrow \mathsf{Setup}(\lambda, \mathsf{LDP})$
$(\hat{m}_0, \hat{m}_1, \hat{\pi}_0, \hat{\pi}_1, \pi_*, \mathsf{c}, \hat{\theta}) \leftarrow \mathcal{A}(\mathsf{set})$
return $(\hat{m}_0 \neq \hat{m}_1 \wedge \mathsf{VerOpenLDP}(\mathsf{c}, \hat{m}_0, \hat{\pi}_0, \hat{\theta})$
    $\wedge\, \mathsf{VerOpenLDP}(\mathsf{c}, \hat{m}_1, \hat{\pi}_1, \hat{\theta}) \wedge \mathsf{VerCommit}(\mathsf{c}, \pi_*))$

$\mathsf{Exp}^{\mathsf{Prob\text{-}LDP\text{-}Binding}}_{P,\mathsf{LDP},\mathcal{A},b}(\lambda)$:
$\mathsf{set} \xleftarrow{\$} \mathsf{Setup}(\lambda, \mathsf{LDP})$
$(\mathsf{c}, \pi_*, m, \pi) \leftarrow \mathcal{A}_1(\mathsf{set})$
$\hat{\theta} \xleftarrow{\$} \Theta$
$(\hat{m}_0, \hat{\pi}) \leftarrow \mathcal{A}_2(\hat{\theta})$
$\hat{m}_1 \leftarrow \mathsf{LDP}(m)$
if $(\mathsf{VerCommit}(\mathsf{c}, \pi_*) \wedge \mathsf{VerOpen}(\mathsf{c}, m, \pi)$
    $\wedge\, \mathsf{VerOpenLDP}(\mathsf{c}, \hat{m}_0, \hat{\pi}, \hat{\theta}) = 1) \wedge (b = 0)$,
then return $\hat{m}_0$, else return $\hat{m}_1$

$\mathsf{OptiGuess}(\lambda, \mathsf{LDP}, \hat{m}, m_0, m_1)$:
$b_* \xleftarrow{\$} \{0, 1\}$
if $\Pr[\hat{m} \leftarrow \mathsf{LDP}(m_0)] > \Pr[\hat{m} \leftarrow \mathsf{LDP}(m_1)]$ then return $0$
if $\Pr[\hat{m} \leftarrow \mathsf{LDP}(m_0)] < \Pr[\hat{m} \leftarrow \mathsf{LDP}(m_1)]$ then return $1$
if $\Pr[\hat{m} \leftarrow \mathsf{LDP}(m_0)] = \Pr[\hat{m} \leftarrow \mathsf{LDP}(m_1)]$ then return $b_*$
```

Figure 3. Security experiments for LDP-C.

LDP mechanism on the original committed message. We stress that in this case the same commitment must open with different values depending on the seed, since the LDP mechanism is probabilistic by nature. We model this property by the experiment Prob-LDP-Binding, where an adversary commits a message m, receives a random seed $\hat{\theta}$, and opens the commitment with $\hat{\theta}$ using the OpenLDP algorithm. Their aim is to return an opened message $\hat{m}$ chosen in a distribution which differs significantly from the distribution generated by the use of the LDP mechanism on m, while guaranteeing that the commitment is correct (*i.e.* the proof π_* is correct), that m is indeed the committed message (*i.e.* the proof π is correct), and that $\hat{m}$ is indeed the message opened by the OpenLDP algorithm using $\hat{\theta}$ as seed (*i.e.* the proof $\hat{\pi}$ is correct). The experiment is parameterized by a bit b. When $b = 0$ and the commitment has been correctly generated and opened by the adversary, the experiment returns the anonymized value opened by the adversary. Otherwise, the experiment itself anonymizes the adversary's message and returns it. Probabilistic-LDP-binding security is achieved when, for any adversary, the message returned follows the same probability distribution, whether $b = 0$ or 1. In other words, the adversary cannot bias the LDP mechanism when it opens the commitment with OpenLDP on a random seed.

Definition 12 (Probabilistic-LDP-Binding)**.** *Let λ be a security parameter,* LDP *be a LDP mechanism, and P be a* LDP-C*. P is said to be* probabilistic-LDP-binding *if for any* PPT *algorithm $\mathcal{A}$ and any message $\hat{m}$, there exists a negligible function ϵ s.t.:*

$$\left|\Pr\left[\hat{m} \leftarrow \mathsf{Exp}^{\mathsf{Prob\text{-}LDP\text{-}Binding}}_{P,\mathsf{LDP},\mathcal{A},0}(\lambda)\right] - \Pr\left[\hat{m} \leftarrow \mathsf{Exp}^{\mathsf{Prob\text{-}LDP\text{-}Binding}}_{P,\mathsf{LDP},\mathcal{A},1}(\lambda)\right]\right| \leq \epsilon(\lambda),$$

the Prob-LDP-Binding *experiment being given in Fig. 3.*

4. LDP Commitment Schemes

In this section, we present an efficient LDP-C scheme for the generalized randomized response mechanism.

4.1. Privacy Parameter for the LDP Mechanism

This LDP mechanism is parameterized by a pair (n_1, n_2). Given a message $m \in \mathcal{M}$, the mechanism returns $\hat{m} = m$ with probability $1/n_1$, otherwise it draws $\hat{m} \xleftarrow{\$} \mathbb{Z}_{n_2}$ and returns $\hat{m}$.

Definition 13 (Generalized Randomized Response Mechanism)**.** *The* generalized randomized response mechanism *parameterized by (n_1, n_2) is the following locally differentially private algorithm:*
$\underline{\mathsf{LDP}(m)}$*: Picks $x \xleftarrow{\$} \mathbb{Z}_{n_1}$, if $x = 0$, then sets $\hat{m} \leftarrow m$, else picks $\hat{m} \xleftarrow{\$} \mathbb{Z}_{n_2}$, finally returns $\hat{m}$.*

We now want to determine how to set the values of n_1 and n_2 given the privacy parameter ϵ and the probabilities of obtaining the correct (resp. an erroneous) value. We have, for any $\hat{m} \neq m$ in $\mathbb{Z}_{n_2}$:

$$\begin{aligned}\Pr[m \leftarrow \mathsf{LDP}(m)] &= \frac{1}{n_1} + \frac{n_1 - 1}{n_1} \cdot \frac{1}{n_2} \\ &= \frac{n_1 + n_2 - 1}{n_1 \cdot n_2} \qquad (Eq.1) \\ \Pr[\hat{m} \leftarrow \mathsf{LDP}(m)] &= \frac{n_1 - 1}{n_1} \cdot \frac{1}{n_2} = \frac{n_1 - 1}{n_1 \cdot n_2}\end{aligned}$$

As LDP implements an optimal staircase mechanism (Section 2), we have the following constraint on the ratio:

$$\left| \ln \frac{\Pr[m \leftarrow \mathsf{LDP}(m)]}{\Pr[\hat{m} \leftarrow \mathsf{LDP}(m)]} \right| = \epsilon$$

so that we can express a relationship between n_1, n_2 and ϵ:

$$n_1 = \frac{n_2 + \exp(\epsilon) - 1}{\exp(\epsilon) - 1}.$$

The parameters are chosen in the following manner. First of all, ϵ is fixed by the user, medical practitioner or medical device producing the value. Recall that the greater the value of ϵ, the lesser the privacy of the mechanism, since the probability of yielding the same outcome as the input value will be higher. In principle, what a user is interested in fixing next, is the actual probability of outputting the correct value, *i.e.* $\Pr[m \leftarrow \mathsf{LDP}(m)]$. With this parameter fixed, using *Eq.1* we can compute:

$$n_1 = \frac{1}{\Pr[m \leftarrow \mathsf{LDP}(m)]} \cdot \frac{\exp(\epsilon)}{\exp(\epsilon) - 1}$$

and deduce n_2. If n_1 is an integer, we can choose to take the floor or ceiling value, depending if we want $\Pr[m \leftarrow \mathsf{LDP}(m)]$ to be a target lower or upper bound.

Example. If we choose $\epsilon = \ln 3$ and $\Pr[m \leftarrow \mathsf{LDP}(m)] = \frac{3}{4}$, this leads us to find $n_1 = n_2 = 2$ which is the classical Randomized Response algorithm [29].

4.2. Naive Solution

We start by giving a naive solution, which is intuitive and correct, but which quickly becomes ineffective when the settings grow larger. This solution uses a standard cryptographic commitment scheme $(\mathsf{Com}, \mathsf{Ope})$, where $\mathsf{Com}(m)$ returns a key k and a committed c, and $\mathsf{Ope}(\mathsf{k}, \mathsf{c})$ returns the committed message m. The general idea is to commit a vector containing the repetition of elements of $\mathbb{Z}_{n_2}$ in such a way that the choice in the uniform distribution over the elements of this vector coincides with the distribution of values returned by the LDP mechanism on a given message. More precisely, given the message m, we define the following vector:

$$\begin{aligned} v &= (v_i)_{i=0}^{n_1 \cdot n_2} \\ &= (\underbrace{0, \cdots, 0}_{n_1 - 1}, \cdots, \underbrace{m-1, \cdots, m-1}_{n_1 - 1}, \underbrace{m, \cdots, m}_{n_1 + n_2 - 1}, \\ &\quad \underbrace{m+1, \cdots, m+1}_{n_1 - 1} \cdots, \underbrace{n_2 - 1, \cdots, n_2 - 1}_{n_1 - 1}).\end{aligned}$$

To commit the message m with the algorithm $\mathsf{Commit}(m, \theta)$ of our naive LDP-C where θ is a permutation chosen from the uniform distribution on the set of the bijective functions $\theta : \mathbb{Z}_{n_1 \cdot n_2} \to \mathbb{Z}_{n_1 \cdot n_2}$, the user runs $(\mathsf{k}_* \mathsf{c}_*) \leftarrow \mathsf{Com}(m)$, runs $(\mathsf{k}_i, \mathsf{c}_i) \leftarrow \mathsf{Com}(v_{\theta(i)})$ for all $i \in \mathbb{Z}_{n_1 \cdot n_2}$, and returns the key $\mathsf{k} = (\mathsf{k}_*, (\mathsf{k}_i)_{i \in \mathbb{Z}_{n_1 \cdot n_2}})$ and the commitment $\mathsf{c} = (\mathsf{c}_*, (\mathsf{c}_i)_{i \in \mathbb{Z}_{n_1 \cdot n_2}})$. The user also proves using zero-knowledge proofs that each possible message is committed at least $n_1 - 1$ times in c and that the value committed in c_* is committed at least $n_1 + n_2 - 1$ times in c to generate the proof π_*. For instance, this proof can easily be built using Schnorr based proofs for discret logarithm relations [30] with the proofs of partial knowledge transformation [31] on Pedersen's commitments [32].

To open the commitment c with the algorithm $\mathsf{Open}(\mathsf{k}, \mathsf{c})$ (*i.e.*, without LDP), the user returns m together with a zero-knowledge proof π that $m = \mathsf{Ope}(\mathsf{k}_*, \mathsf{c}_*)$ without revealing k_*. This proof can easily be built using Schnorr based proofs for discret logarithm relations on Pedersen's commitments.

Given a random seed $\hat{\theta}$ chosen from $\mathbb{Z}_{n_1 \cdot n_2}$, the user can also open the commitment c by applying the LDP mechanism with the algorithm $\mathsf{OpenLDP}(\mathsf{k}, \mathsf{c}, \hat{\theta})$. To do this, the user returns the committed message $\hat{m}$ in $\mathsf{c}_{\hat{\theta}}$ and a proof $\hat{\pi}$ that $\hat{m} = \mathsf{Ope}(\mathsf{k}_{\hat{\theta}}, \mathsf{c}_{\hat{\theta}})$ without revealing $\mathsf{k}_{\hat{\theta}}$ (using, once again, Schnorr based proofs for discret logarithm relations on Pedersen's commitments).

The hiding and LDP-hiding properties derive from the hiding property of the commitment scheme used and from the fact that the proofs are zero-knowledge. The use of permutation ensures that the verifier cannot use the choice of $\hat{\theta}$ to determine whether the value $\hat{m}$ is the original message or another message; $\hat{m}$ provides as much information as a randomly chosen value v_i in v, and therefore as much information as a message that is actually anonymized with the generalized randomized response LDP mechanism. Binding and LDP-binding result directly from the binding property of the cryptographic commitment scheme used. The probabilistic-LDP-binding is ensured by the soundness of the zero-knowledge proofs, which guarantee that if $\hat{\theta}$ is indeed random, then the opening of the commitment $\mathsf{c}_{\hat{\theta}}$ will follow the distribution of the LDP mechanism, since the c_i are commitments of the values of the permuted vector v.

The size of the signatures and the computational complexity of the commitments and verifications are linear in $n_1 \cdot n_2$, which seems unsatisfactory, especially when we want to use large parameters.

4.3. Efficient Scheme with Logarithmic Commitments

We will now present a more efficient LDP-C scheme called Optimized Randomized Response Commitment (ORRC) whose complexity in computation time and in the size of commitments, proofs, and openings is at most

$O(\max(\log_2(n_1), \log_2(n_2)))$. Our scheme uses parameters n_1 and n_2 which are powers of 2, so $n_1 = 2^{\ell_1}$ and $n_2 = 2^{\ell_2}$. In practical terms, this means that the messages are chosen from vectors of ℓ_2 bits. Moreover, for fixed ℓ_2 and ϵ, we will choose the smallest ℓ_1 s.t. $\ell_1 \geq \log_2\left(\frac{2^{\ell_2}+\exp(\epsilon)-1}{\exp(\epsilon)-1}\right)$.

Definition 14 (ORRC scheme). *Let (ℓ_1, ℓ_2) be two integers. The* optimized randomized response commitment *scheme ORRC is a* LDP-C *scheme depending on the generalized randomized response LDP mechanism parameterized by $(2^{\ell_1}, 2^{\ell_2})$ consisting of algorithms* (Setup, Commit, Open, OpenLDP, VerOpen, VerOpenLDP, VerCommit) *described in the rest of this section.*

The setup algorithm generates a group of prime order and several elements of this group. More precisely, we generate pairs $(f_{i,0}, f_{i,1})$, $(g_{i,0}, g_{i,1})$, and $(h_{i,0}, h_{i,1})$, which will be used to commit bit strings. To do this, given an integer ℓ and a bit string $s \in \{0,1\}^\ell$, and using the pairs $(g_{i,0}, g_{i,1})$, we choose a random element x, set $y = g^x$, and compute $S_i = g_{i,s[i]}$ for all i in $[\![\ell]\!]$. The opening of the commitment consists in revealing x. In this way, we can simply prove, using proofs of equality of discrete logarithms, that the committed message is indeed an element of $\{0,1\}^\ell$; it suffices to show, for all i, that $S_i = g_{i,0}$ or $S_i = g_{i,1}$.

Note that it is also possible to check the commitment quite efficiently by checking $\prod_{i\in[\![\ell]\!]} S_i^x = \left(\prod_{i\in[\![\ell]\!]} g_{i,s[i]}\right)^x$ and $y = g^x$, rather than by checking each of the S_i independently. This is because exponentiation in a group of prime order is much more expensive than multiplication, so it is much more efficient to have a constant number of exponentiations that do not depend on ℓ, even with a linear number of multiplications.

<u>Setup(λ, LDP):</u> generates a group $\mathbb{G} = \langle g \rangle$ of prime order p, for each $i \in [\![\ell_1]\!]$ and $j \in \{0,1\}$, picks $f_{i,j} \xleftarrow{\$} \mathbb{G}$, then for each $i \in [\![\ell_2]\!]$ and each $j \in \{0,1\}$, picks $g_{i,j}, h_{i,j} \xleftarrow{\$} \mathbb{G}$. It sets $\mathcal{M} \leftarrow \{0,1\}^{\ell_2}$ and $\Theta \leftarrow \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2}$, and returns set $\leftarrow (\lambda, \mathbb{G}, p, (g_{i,j})_{\substack{i\in[\![\ell_1]\!] \\ j\in\{0,1\}}}, (f_{i,j}, h_{i,j})_{\substack{i\in[\![\ell_2]\!] \\ j\in\{0,1\}}}, \mathcal{M}, \Theta)$.

The idea behind our scheme is as follows. The seed θ is a pair (s, t) chosen in the uniform distribution on $\Theta = \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2}$ by the user. To commit m with ORRC, we commit s, m, and t in the respective commitments A_1, A_2 and B (using the method described above). To open without LDP, we simply open A_2. To open with the LDP mechanism, given a seed $\hat{\theta} \in \Theta$ parsed as $(\hat{s}, \hat{t})$ chosen by the verifier, we test whether $s = \hat{s}$ (which happens with a probability of $1/2^{\ell_1}$). If so, we open the actual message m from A_2, otherwise we return the message $\hat{m} = t \oplus \hat{t}$ with proof that $\hat{m}$ has been computed correctly with the value t committed to B (we stress that if t or $\hat{t}$ was indeed chosen randomly, the message $\hat{m}$ will follow a uniform distribution over $\mathcal{M}$). In addition, we use a zero-knowledge proof to prove that the correct operation between the two possible has been performed, depending on whether $s = \hat{s}$ has been performed, and without revealing s.

<u>Commit(m, θ):</u> picks $x \in \mathbb{Z}_p^*$, sets $y = g^x$, and parses θ as (s, t). For all $i \in [\![\ell_1]\!]$, it sets $A_{(i,1)} \leftarrow g^x_{i,s[i]}$. For all $i \in [\![\ell_2]\!]$, it sets $(A_{(i,2)}, B_{(i,0)}, B_{(i,1)}) \leftarrow (f^x_{i,m[i]}, h^x_{i,t[i]}, h^x_{i,1\oplus t[i]})$. It then computes:

$$\pi_{A_1} \leftarrow \mathsf{NIP}\left\{x : \bigwedge_{i=1}^{\ell_1} \left(\begin{array}{c} y = g^x \\ \wedge \left(\bigvee_{j=0}^1 A_{(i,1)} = g^x_{i,j}\right) \end{array} \right)\right\}$$

$$\pi_{A_2} \leftarrow \mathsf{NIP}\left\{x : \bigwedge_{i=1}^{\ell_2} \left(\begin{array}{c} y = g^x \\ \wedge \left(\bigvee_{j=0}^1 A_{(i,2)} = f^x_{i,j}\right) \end{array} \right)\right\}$$

$$\pi_B \leftarrow \mathsf{NIP}\left\{x : \bigwedge_{i=1}^{\ell_2} \left(\begin{array}{c} y = g^x \\ \wedge \left(\bigwedge_{j=0}^1 B_{(i,j)} = h^x_{i,j}\right. \\ \left.\vee \bigwedge_{j=0}^1 B_{(i,j)} = h^x_{i,1-j}\right) \end{array} \right)\right\}.$$

It sets $\mathsf{k} \leftarrow x, \mathsf{c} \leftarrow (y, (A_{(i,1)})_{i\in[\![\ell_1]\!]}, (A_{(i,2)}, B_{(i,0)}, B_{(i,1)})_{i\in[\![\ell_2]\!]}), \pi_* \leftarrow (\pi_{A_1}, \pi_{A_2}, \pi_B)$. Finally, it returns $(\mathsf{k}, \mathsf{c}, \pi_*)$.

<u>VerCommit(c, π_*):</u> parses c as $(y, (A_{(i,1)})_{i\in[\![\ell_1]\!]}, (A_{(i,2)}, B_{(i,0)}, B_{(i,1)})_{i\in[\![\ell_2]\!]})$ and π_* as $(\pi_{A_1}, \pi_{A_2}, \pi_B)$, verifies the proofs in π_*, and returns 1 if the proof is valid, 0 otherwise.

Note that $B_{(i,0)}$ commits the bits of t and $B_{(i,1)}$ commits the bits of $\bar{t}$, we will see later that these elements will be useful for showing that in the case $s \neq \hat{s}$, $\hat{m} = t \oplus \hat{t}$. The opening algorithm with no LDP consists of opening the $A_{(i,2)}$ in which m is committed, using the method given above.

<u>Open(k, c):</u> parses k as x and c as $(y, (A_{(i,1)})_{i\in[\![\ell_1]\!]}, (A_{(i,2)}, B_{(i,0)}, B_{(i,1)})_{i\in[\![\ell_2]\!]})$. For each $i \in [\![\ell_2]\!]$, it sets $m[i] = b$ iff $A_{(i,2)} = f^x_{i,b}$ where $b \in \{0,1\}$. It computes $A_2 = \prod_{i=1}^{\ell_2} A_{(i,2)}$ and $\alpha_2 = \prod_{i=1}^{\ell_2} f_{i,m[i]}$. It computes the proof $\pi \leftarrow \mathsf{NIP}\{x : y = g^x \wedge A_2 = \alpha_2^x\}$ and returns m and π.

<u>VerOpen(c, m, π):</u> parses c as $(y, (A_{(i,1)})_{i\in[\![\ell_1]\!]}, (A_{(i,2)}, B_{(i,0)}, B_{(i,1)})_{i\in[\![\ell_2]\!]})$, computes A_2 and α_2 as in the algorithm Open and verifies the proof π.

As we already mentioned, to open the commitment with the LDP mechanism on the message $\hat{m}$ and the seed $\hat{\theta} = (\hat{s}, \hat{t})$, we show that if s, the value committed to $A_{(i,1)}$, is equal to $\hat{s}$, then the value committed to $A_{(i,2)}$ is $\hat{m}$, otherwise we show that $\hat{m} = t \oplus \hat{t}$ where t is the value committed to $B_{(i,0)}$. We will now focus on the second case. Recall that for all i, $B_{(i,0)}$ commits $t[i]$ and $B_{(i,1)}$ commits $1 \oplus t[i]$. We remark that:

$$\begin{aligned} h^x_{i,\hat{t}[i]} &= \begin{cases} h^x_{i,t[i]} & \text{if} \quad t[i] = \hat{t}[i], \\ h^x_{i,1\oplus t[i]} & \text{else} \quad (t[i] \neq \hat{t}[i]). \end{cases} \\ &= \begin{cases} B_{(i,0)} & \text{if} \quad t[i] = \hat{t}[i], \\ B_{(i,1)} & \text{else.} \end{cases} \\ &= B_{(i,(t\oplus\hat{t})[i])} \\ &= B_{(i,\hat{m}[i])}. \end{aligned}$$

Therefore, to show that $m = t \oplus \hat{t}$, it is sufficient to prove that $\left(\prod_{i\in[\![\ell_2]\!]} h_{i,\hat{t}[i]}\right)^x = \prod_{i\in[\![\ell_2]\!]} B_{(i,\hat{m}[i])}$ and $y = g^x$. This results in the following opening algorithm.

<u>OpenLDP(k, c, $\hat{\theta}$):</u> parse k as x, c as $(y, (A_{(i,1)})_{i\in[\![\ell_1]\!]}, (A_{(i,2)}, B_{(i,0)}, B_{(i,1)})_{i\in[\![\ell_2]\!]})$ and $\hat{\theta}$ as $(\hat{s}, \hat{t})$. It sets:

$$A_1 = \prod_{i=1}^{\ell_1} A_{(i,1)};\; A_2 = \prod_{i=1}^{\ell_2} A_{(i,2)};\; \alpha_1 = \prod_{i=1}^{\ell_1} g_{i,\hat{s}[i]};$$

If $A_1 = \alpha_1^x$ it sets $\hat{m} \leftarrow m$, else $\hat{m} \leftarrow t \oplus \hat{t}$. It sets:

$$\alpha_2 = \prod_{i=1}^{\ell_2} f_{i,\hat{m}[i]}; \quad \beta = \prod_{i=1}^{\ell_2} B_{(i,\hat{m}[i])}; \ \gamma = \prod_{i=1}^{\ell_2} h_{i,\hat{t}[i]},$$

then it generates the following proof:

$$\hat{\pi} \leftarrow \mathsf{NIP}\left\{ x : \begin{array}{c} (y = g^x \wedge A_1 = \alpha_1^x \wedge A_2 = \alpha_2^x) \\ \vee (y = g^x \wedge A_1 \neq \alpha_1^x \wedge \beta = \gamma^x) \end{array} \right\}.$$

Finally, it returns $(\hat{m}, \hat{\pi})$.

$\mathsf{VerOpenLDP}(\mathsf{c}, \hat{m}, \hat{\pi}, \hat{\theta})$**:** parses $\hat{\theta}$ as $(\hat{s}, \hat{t})$ and c as $(y, (A_{(i,1)})_{i \in [\![\ell_1]\!]}, (A_{(i,2)}, B_{(i,0)}, B_{(i,1)})_{i \in [\![\ell_2]\!]})$, then computes A_1, A_2, α_1, α_2, β, and γ as in the algorithm $\mathsf{OpenLDP}$. It verifies the proof $\hat{\pi}$, and returns 1 if the proof is valid, 0 otherwise.

In Appendix H, we prove the correctness of our scheme. In Appendix B, we give three possible extensions of our scheme, (i) allowing to open multiple commitments, (ii) to choose a larger ϵ a posteriori, and (iii) to choose a parameter n_1 that is not necessarily a power of 2 (which allows to choose ϵ more accurately for a given set of values).

4.4. Security Analysis

Theorem 3. *The* ORRC *scheme instantiated with extractable and zero-nowledge proofs and a group where the DDH assumption holds is hiding, LDP-hiding, binding, LDP-binding and probabilistic-LDP-binding.*

In the following, we provide sketches of the security proofs, giving the sequence of games [28] used for the proof, but omitting the reductions between the games. The full proofs are available in Appendix J.

The hiding proofs requires the two following assumptions in the chosen group $\mathbb{G} = \langle g \rangle$ of prime order p:

Left-or-Right-DDH (LOR-DDH)**.** the advantage of any PPT algorithm $\mathcal{A}$ that receives random elements g^x, g^{y_0}, g^{y_1}, and g^{xy_b} for a random bit b and that tries to guess b is bounded by a negligible function $\epsilon_{\mathsf{LOR\text{-}DDH}}(\lambda)$.

Left-or-Right-DDH-2 ($\mathsf{LOR\text{-}DDH}-2$)**.** the advantage of any PPT algorithm $\mathcal{A}$ that receives random elements g^x, g^{y_0}, g^{y_1}, and g^{xy_b} and $g^{xy_{1-b}}$ for a random bit b and that tries to guess b is bounded by a negligible function $\epsilon_{\mathsf{LOR\text{-}DDH-2}}(\lambda)$.

We prove in Appendix G that these two assumptions reduce to DDH (Lemmas 2 and 3).

Proof sketch that ORRC *is hiding.* We use the following sequence of games:

Game G_0: This game is the same as the Hiding experiment in the case where $b = 0$.

Game G_1: This game is the same as the game G_0 except that the challenger replaces the zero-knowledge proof π_* by a simulated proof, we have:

$$\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_0] = \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_1].$$

Game G_2: This game is the same as the Hiding experiment in the case where $b = 1$. In the game G_1 (resp. G_2), the challenger commits the message m_0 (resp. m_1), so the i-th bit of m_0 (resp. m_1) is contained in the part $A_{(i,2)}$ of c. To prove indistinguishability between G_1 and G_2, we consider the following hybrid argument [33]:

Game $G_{1,k}$ (for all $k \in [\![\ell_2]\!]$): We define $G_{1,0}$ as G_1 and G_{1,ℓ_2} as G_2. This game is the same as the game $G_{1,k-1}$ except that the challenger replaces $A_{(k,2)} = f^x_{k,m_0[k]}$ by $f^x_{k,m_1[k]}$ in the commitment. By reduction we can show that for all $k \in [\![\ell_2]\!]$:

$$|\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1}] - \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k}]| \\ \leq \epsilon_{\mathsf{LOR\text{-}DDH}}(\lambda).$$

This leads to the advantage $\epsilon(\lambda) = \ell_2 \cdot \epsilon_{\mathsf{LOR\text{-}DDH}}(\lambda)$ for the hiding property, which concludes the proof. □

Proof sketch that ORRC *is ROS-LDP-hiding.* We define the algorithm LDP for some value $\hat{s} \in [\![\ell_1]\!]$ and the simulator as follows:

$\mathsf{LDP}(m)$: picks $(s, t) \xleftarrow{\$} \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2}$, if $s = 0$, then sets $\hat{m} = m$, else, sets $\hat{m} = t$ and returns $\hat{m}$.

$\mathsf{Sim}(\hat{m})$: uses the zero-knowledge proof simulator to produce $\hat{\pi}$.

We use the following sequence of games:

Game G_0: This game is the same as the $\mathsf{ROS\text{-}LDP\text{-}Hiding}$ experiment in the case $b = 0$.

Game G_1: This game is the same as the game G_0 except that the challenger replaces the proofs $\hat{\pi}$ and π_* by simulated proofs. The loss of advantage between the games is the same as between G_0 and G_1 in the previous proof.

From the game G_1, we parse the θ used to produce c as (s_0, t_0). We note that if $s_0 = \hat{s}$, then $\hat{m} = m$, else $\hat{m} = t_0 \oplus \hat{t}$. In what follows, we will gradually replace (s_0, t_0) (bit by bit) by other values (s_1, t_1) chosen at random in the commitment c, but will continue to compute $\hat{m}$ with (s_0, t_0). This leads us to the $\mathsf{ROS\text{-}LDP\text{-}Hiding}$ experiment in the case $b = 1$.

Game G_2: This game is the same as the game G_1 except that for all $i \in [\![\ell_1]\!]$, the challenger replaces $A_{i,1} = g^x_{i,s_0[i]}$ by $g^x_{i,s_1[i]}$. To prove indistinguishability between G_1 and G_2, we consider the following hybrid argument [33]:

Game $G_{1,k}$(for all $k \in [\![\ell_1]\!]$): We define $G_{1,0}$ as G_1 and G_{1,ℓ_1} as G_2. This game is the same as the game $G_{1,k-1}$ except that the challenger replaces $A_{(k,1)} = g^x_{k,s_0[k]}$ by $g^x_{k,s_1[k]}$. By reduction we can show that for all $k \in [\![\ell_1]\!]$:

$$|\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1}] - \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k}]| \\ \leq \epsilon_{\mathsf{LOR\text{-}DDH}}(\lambda).$$

Game G_3: This game is the same as the game G_2 except that for all $i \in [\![\ell_2]\!]$, the challenger replaces $\left(B_{(i,0)}, B_{(i,1)}\right) = \left(h^x_{i,t_0[i]}, h^x_{i,1\oplus t_0[i]}\right)$ by $\left(h^x_{i,t_1[i]}, h^x_{i,1\oplus t_1[i]}\right)$. To prove indistinguishability between G_2 and G_3, we consider the following hybrid argument [33]:

Game $G_{2,k}$(for all $k \in [\![\ell_2]\!]$): We define $G_{2,0}$ as G_2 and G_{2,ℓ_2} as G_3. This game is the same as the game $G_{2,k-1}$ except that the challenger replaces $(B_{(k,0)}, B_{(k,1)}) = (h^x_{k,t_0[k]}, h^x_{k,1\oplus t_0[k]})$ by $(h^x_{k,t_1[k]}, h^x_{k,1\oplus t_1[k]})$. By reduction we can show that for all $k \in [\![\ell_2]\!]$:

$$|\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k-1}] - \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k}]| \\ \leq \epsilon_{\mathsf{LOR\text{-}DDH-2}}(\lambda).$$

This leads to the advantage $\epsilon(\lambda) = \ell_1 \cdot \epsilon_{\mathsf{LOR\text{-}DDH}}(\lambda) + \ell_2 \cdot \epsilon_{\mathsf{LOR\text{-}DDH-2}}(\lambda)$ for the hiding property, which concludes the proof. □

According to Theorem 2, ORRC is IND-LDP-hiding because it is hiding and ROS-LDP-hiding.

The hiding proofs requires the following assumptions in the chosen group $\mathbb{G} = \langle g \rangle$ of prime order p:
ℓ-product-collision-resistance (ℓ-col). the probability of any PPT algorithm $\mathcal{A}$ that receives random elements $\left(g_{(i,j)}\right)_{i \in [\![\ell]\!]; j \in \{0,1\}}$ to find some (x_1, x_2) s.t. $x_1 \neq x_2$ and $\prod_{i=1}^{\ell} g_{(i,x_1[i])} = \prod_{i=1}^{\ell} g_{(i,x_2[i])}$ is bounded by a negligible function $\epsilon_{\ell\text{-col}}(\lambda)$. We prove in Appendix G that this assumption reduces to the discrete logarithm assumption (Lemma 4). (Lemma 4).

Proof sketch that ORRC is binding. We use the following sequence of games:
Game G_0: This game is the same as the Binding experiment.
Game G_1: This game is the same as the game G_0 except that the challenger uses the extractors on the zero-knowledge proofs π_0, π_1 outputted by the adversary, and aborts and returns 0 on the event F_1 = "An extractor fails on at least one proof". The loss of advantage between the games is the same as between G_0 and G_1 in the previous proof. We emphasize that if $\mathcal{A}$ wins its game, then the witnesses x of all proofs are correctly extracted and are the same according to the proofs structure. We have:

$$|\Pr[\mathcal{A} \text{ wins } G_0] - \Pr[\mathcal{A} \text{ wins } G_1]| \leq \Pr[F_1] \\ \leq 2 \cdot \epsilon_{\mathsf{ext}}(\lambda).$$

Game G_2: This game is the same as the game G_0 except that aborts and returns 0 on the event F_2 = "$\mathcal{A}$ outputs $(m_0, m_1, \pi_0, \pi_1, \mathsf{c})$ s.t. $\prod_{i=1}^{\ell_2} f_{i,m_0[i]} = \prod_{i=1}^{\ell_2} f_{i,m_1[i]}$ and $m_0 \neq m_1$", we have:

$$|\Pr[\mathcal{A} \text{ wins } G_1] = \Pr[\mathcal{A} \text{ wins } G_2]| \leq \Pr[F_2].$$

By reduction we can show that $\Pr[F_2] \leq \epsilon_{\ell_2-\mathrm{col}}(\lambda)$. If F_1 and F_2 does not happen, then the values $A_{(i,2)}$ are correct and there is no product collision for two different messages, so $m_0 = m_1$ and so $\Pr[\mathcal{A} \text{ wins } G_2] = 0$. This leads to the advantage $\epsilon(\lambda) = 2 \cdot \epsilon_{\mathsf{ext}}(\lambda) + \epsilon_{\ell_2-\mathrm{col}}(\lambda)$ for the binding property, which concludes the proof. □

Proof sketch that ORRC is LDP-binding. We use the following sequence of games:
Game G_0: This game is the same as the LDP-Binding experiment.
Game G_1: This game hop is similar to the hop between G_0 and G_1 in the previous proof except that we use the extractor on the proofs $\hat{\pi}_0$, $\hat{\pi}_1$ and $\pi_* = (\pi_{A_1}, \pi_{A_2}, \pi_B)$.
Game G_2: This game is the same as the game G_1 except that aborts and returns 0 on the event F_2 = "$\mathcal{A}$ returns $(\hat{m}_0, \hat{m}_1)$ s.t. $\prod_{i=1}^{\ell_2} f_{i,\hat{m}_0[i]} = \prod_{i=1}^{\ell_2} f_{i,\hat{m}_1[i]}$ and $\hat{m}_0 \neq \hat{m}_1$". we have:

$$|\Pr[\mathcal{A} \text{ wins } G_1] = \Pr[\mathcal{A} \text{ wins } G_2]| \leq \Pr[F_2].$$

By reduction we can show that $\Pr[F_2] \leq \epsilon_{\ell_2\text{-col}}(\lambda)$.
Note that in the case where $\hat{s} \neq s$, we have $h^x_{i,\hat{t}[i]} = h^x_{i,\hat{m}[i] \oplus t[i]}$.
Game G_3: This game is the same as the game G_2 except that aborts and returns 0 on the event F_3 = "$\mathcal{A}$ returns $(\hat{m}_0, \hat{m}_1)$ s.t. $\prod_{i=1}^{\ell_2} h_{i,\hat{m}_0[i] \oplus t[i]} = \prod_{i=1}^{\ell_2} h_{i,\hat{m}_1[i] \oplus t[i]}$ and $\hat{m}_0 \neq \hat{m}_1$ and $s \neq \hat{s}$", we have:

$$|\Pr[\mathcal{A} \text{ wins } G_2] = \Pr[\mathcal{A} \text{ wins } G_3]| \leq \Pr[F_3].$$

By reduction we can show that $\Pr[F_3] \leq \epsilon_{\ell_2\text{-col}}(\lambda)$.
Game G_4: This game is the same as the game G_3 except that aborts and returns 0 on the event F_4 = "$\mathcal{A}$ returns $(\hat{m}_0, \hat{m}_1)$ s.t. $\prod_{i=1}^{\ell_2} B_{(i,\hat{m}_0[i])} = \prod_{i=1}^{\ell_2} B_{(i,\hat{m}_1[i])}$ and $\hat{m}_0 \neq \hat{m}_1$ and $s \neq \hat{s}$", we have:

$$|\Pr[\mathcal{A} \text{ wins } G_3] = \Pr[\mathcal{A} \text{ wins } G_4]| \leq \Pr[F_4].$$

By reduction we can show that $\Pr[F_4] \leq \epsilon_{\ell_2\text{-col}}(\lambda)$.
If F_1, F_2, F_3, and F_4 does not happen, then the commitment is well formed and there is no product collision for two different messages on the α_2, β, and γ, which implies the uniqueness of the open message for $(\hat{s}, \hat{t})$, so $m_0 = m_1$ and so $\Pr[\mathcal{A} \text{ wins } G_4] = 0$. This leads to the advantage $\epsilon(\lambda) = 5 \cdot \epsilon_{\mathsf{ext}}(\lambda) + 3 \cdot \epsilon_{\ell_2-\mathrm{col}}(\lambda)$ for the LDP-binding property, which concludes the proof. □

Proof sketch that ORRC is probabilistic-LDP-binding. We use the following sequence of games:
Game G_0: This game is the same as the Prob-LDP-Binding experiment in the case where $b = 0$.
Game G_1: This game hop is similar to the hop between G_0 and G_1 in the previous proof.
Game G_2: This game is the same as the game G_1 except that the challenger aborts and returns $\hat{m} \xleftarrow{\$} \mathcal{M}$ on the event F_2 = "$\mathcal{A}_1$ returns $(\mathsf{c}, \pi_*, m, \pi)$ and $\mathcal{A}_2$ returns $(\hat{m}_0, \hat{\pi})$ s.t. $\prod_{i=1}^{\ell_2} f_{i,\hat{m}_0[i]} = \prod_{i=1}^{\ell_2} f_{i,m[i]}$ and $\hat{m}_0 \neq m$", we have that $\forall \hat{m} \in \mathcal{M}$:

$$|\Pr[\hat{m} \leftarrow G_1] - \Pr[\hat{m} \leftarrow G_2]| \leq \Pr[F_2].$$

By reduction we can show that $\Pr[F_2] \leq \epsilon_{\ell_2\text{-col}}(\lambda)$.
Note that in the case where $\hat{s} \neq s$, we have $\hat{m} = t \oplus \hat{t}$, which implies $h^x_{i,\hat{t}[i]} = h^x_{i,\hat{m}[i] \oplus t[i]}$ and $B_{(i,\hat{m}[i])} = h^x_{i,\hat{t}[i]}$.
Game G_3: This game is the same as the game G_3 except that aborts and returns $\hat{m} \xleftarrow{\$} \mathcal{M}$ on the event F_3 = "$\mathcal{A}_1$ returns $(\mathsf{c}, \pi_*, m, \pi)$ and $\mathcal{A}_2$ returns $(\hat{m}_0, \hat{\pi})$ s.t. $\prod_{i=1}^{\ell_2} h_{i,\hat{m}_0[i] \oplus t[i]} = \prod_{i=1}^{\ell_2} h_{i,\hat{t}[i]}$ and $\hat{m}_0 \oplus t \neq \hat{t}$ and $s \neq \hat{s}$", we have that, $\forall \hat{m} \in \mathcal{M}$:

$$|\Pr[\hat{m} \leftarrow G_3] - \Pr[\hat{m} \leftarrow G_4]| \leq \Pr[F_3].$$

By reduction we can show that $\Pr[F_3] \leq \epsilon_{\ell_2\text{-col}}(\lambda)$.
Game G_4: This game is the same as the game G_3 except that aborts and returns $\hat{m} \xleftarrow{\$} \mathcal{M}$ on the event F_4 = "$\mathcal{A}_1$ returns $(\mathsf{c}, \pi_*, m, \pi)$ s.t. $\prod_{i=1}^{\ell_1} g_{i,s[i]} = \prod_{i=1}^{\ell_1} g_{i,\hat{s}[i]}$ and $s \neq \hat{s}$", we have that, $\forall \hat{m} \in \mathcal{M}$:

$$|\Pr[\hat{m} \leftarrow G_3] - \Pr[\hat{m} \leftarrow G_4]| \leq \Pr[F_4].$$

By reduction we can show that $\Pr[F_4] \leq \epsilon_{\ell_1\text{-col}}(\lambda)$.
If F_1, F_2, F_3, and F_4 does not happen, then the commitment is well formed and the only way to open the commitment is to return m if $s = \hat{s}$, and to return $t \oplus \hat{t}$ otherwise. We deduce that G_4 is the same as the Prob-LDP-Binding experiment in the case where $b = 1$. This leads to the advantage $\epsilon(\lambda) = 5 \cdot \epsilon_{\mathsf{ext}}(\lambda) + \epsilon_{\ell_1-\mathrm{col}}(\lambda) + 2 \cdot \epsilon_{\ell_2-\mathrm{col}}(\lambda)$ for the probabilistic-LDP-binding property, which concludes the proof. □

5. Instantiation and Implementation

ORRC in our use case. We next detail the use of the ORRC algorithms in the application suggested in the introduction and summarised in Fig. 1. During interaction 1a-b), Diana (who has a pair $(\mathsf{pk}, \mathsf{sk})$ of signature keys) agrees with her patients to choose the privacy parameter ϵ. Diana (or the connected medical device) then retrieves the patient data. For each data m, Diana chooses the parameters (ℓ_1, ℓ_2) according to the size of the set of possible values of the data and the patient's ϵ (Note that she can use the same setup set as long as it has been generated from large enough parameters, i.e. a higher bound on ℓ_1 and ℓ_2). Diana (or some device acting on her behalf) commits the data using a random seed θ by running $(\mathsf{k}, \mathsf{c}, \pi_*) \leftarrow \mathsf{Commit}(m, \theta)$. Diana then signs each (or a batch of) committed data $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mathsf{c})$, with possibly additional information describing the nature of the data. Finally, Diana sends the elements $(\epsilon, m, \mathsf{k}, \mathsf{c}, \pi_*, \sigma)$ to Helen during interaction 2a-b). When Robin requests data from Helen, he sends his request with a random seed $\hat{\theta}$. Helen runs $(\hat{m}, \hat{\pi}) \leftarrow \mathsf{OpenLDP}(\mathsf{k}, \mathsf{c}, \hat{\theta})$ and sends $(\epsilon, \hat{m}, \mathsf{c}, \pi_*, \sigma, \hat{\pi})$ to Robin during interaction 3). Robin verifies π_*, σ, and $\hat{\pi}$, and uses the data $\hat{m}$ for his open science research. In case of an audit, during interaction 4a) Robin forwards $(\epsilon, \hat{m}, \mathsf{c}, \pi_*, \sigma, \hat{\pi})$ to Arthur, who can in turn verify that the data has been correctly anonymized. Finally, Robin can forward the same values in interaction 4b) to convince the reviewer Rachel that the data used in his study are relevant.

Instantiation of the NIP. All the NIP languages used in our scheme correspond to boolean relations of equality and inequality of discrete logarithms. These proofs can be instantiated as sigma protocols, where the prover sends a commitment, receives a challenge, and returns a response. By using the hash (produced by a random oracle) of the statement and the commitment as a challenge, these proofs become non-interactive [34]. Such a proof for the equality of discrete logarithms is given in [30], and a proof for the inequality is given in [35]. To prove that several statements are true at the same time, it is sufficient to use the same challenge for all the proofs. To prove that a single statement is true among several, we can use the transformation given in [31]. The proofs produced in this way are linear in the number of statements (in size and time). These proofs are zero-knowledge and extractable, so we can use them to instantiate our construction. In Appendix C, we give full details of the construction of the proofs required for ORRC as well as an evaluation of their asymptotic complexity in terms of computational cost and size.

Instantiation of the signature. Remember that in our application, the LDP commitment needs to be signed. The data is committed by a user, and will be opened (with LDP) by a delegate, who must be able to convince a verifier that the data it receives is indeed the data committed by the user after the LDP mechanism has been applied. From a formal point of view, a signature is a triplet of algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$ s.t. $\mathsf{Gen}(\lambda)$ returns a private/public key pair$(\mathsf{pk}, \mathsf{sk})$, $\mathsf{Sign}(\mathsf{sk}, m)$ returns a signature σ for message m, and $\mathsf{Ver}(\mathsf{pk}, m)$ decides whether a signature is valid or not. A signature is EUF-CMA if no polynomial adversary is able, given the public key pk and an oracle that produces signatures $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$ for chosen messages m, to forge a valid and fresh signature (*i.e.* one that has not been produced by the oracle). In our implementation, we use the well-known Schnorr's signature [36].

Evaluation of the asymptotic complexity. In Table 1 we give the complexity in terms of computation time for our naive solution and for ORRC depending on the parameters (ℓ_1, ℓ_2). For the sake of clarity, we give the complexity in terms of the number of exponentiations, since the multiplication cost is negligible in comparison, but we indicate when the complexity in terms of the number of multiplications is different. In Table 2 we give the complexity in terms of size. The size is evaluated in number of group elements. These results are based on the evaluation of the complexity of the NIP used in our schemes, which is available in Appendix C.

TABLE 1. COMPUTATION COST COMPLEXITY ANALYSIS FOR OUR SCHEMES.
$(*)$ requires $O(\max(\ell_1, \ell_2))$ random draws in the group.
$(**)$ requires $O(\max(\ell_1, \ell_2))$ group operations.

Scheme	Naive	ORRC
Setup	$O(1)$	$O(1)^*$
Commit	$O(2^{\ell_1+2\cdot\ell_2})$	$O(\max(\ell_1, \ell_2))$
VerCommit	$O(2^{\ell_1+2\cdot\ell_2})$	$O(\max(\ell_1, \ell_2))$
Open	$O(1)$	$O(\max(\ell_1, \ell_2))$
VerOpen	$O(1)$	$O(1)^{**}$
OpenLDP	$O(1)$	$O(1)^{**}$
VerOpenLDP	$O(1)$	$O(1)^{**}$

TABLE 2. SIZE COMPLEXITY ANALYSIS FOR OUR SCHEMES.

Scheme	Naive	ORRC
set	$O(1)$	$O(\max(\ell_1, \ell_2))$
k	$O(2^{\ell_1+\ell_2})$	$O(1)$
c	$O(2^{\ell_1+2\cdot\ell_2})$	$O(\max(\ell_1, \ell_2))$
π_*	$O(2^{\ell_1+2\cdot\ell_2})$	$O(\max(\ell_1, \ell_2))$
π	$O(1)$	$O(1)$
$\hat{\pi}$	$O(1)$	$O(1)$
θ	$O(2^{\ell_1+\ell_2})$	$O(1)$
$\hat{\theta}$	$O(1)$	$O(1)$

Implementation. To evaluate the efficiency of our LDP-C schemes, we implemented the algorithm Setup, Commit, VerCommit, Open, VerOpen, OpenLDP and VerOpenLDP for both our naive solution and ORRC in Rust on a processor 11th Gen Intel® Core™ i7-1185G7 @ 3.00GHz × 8. The source code is available at [37].

We used the Ristretto prime order group and the curve25519-dalek [38] library, with 255 bits secret keys (full dependencies of our implementation are listed in the file `cargo.toml`.). We also implemented a Schnorr Signature scheme constituted by the algorithms Gen, Sign and Ver to sign the generated commitment. For ORRC, we measured the execution time of each algorithm over 1000 runs. The results are summarized in Table 3 for the runtime and in Table 4 for the size (a graphical representation of these results is given in Appendix D); They depend on the length of the parameters ℓ_1 and ℓ_2 thus for message space sizes $n_2 = 2^{\ell_2}$ ranging from 16 to 10^9 and ϵ around 0.70 (between 0.69 and 0.72). We emphasize that our solution is particularly efficient for the verifier of openings, as we had suspected. The commitment correction verification takes a little longer, but it can be pre-computed before the commitment is opened, since it only concerns the commitment and not its opening. A similar analysis for the naive solution, as well as a comparison, are given in Appendix E.

TABLE 3. RUNNING TIME IN MILLISECONDS OF OUR ALGORITHMS FOR THE ORRC SCHEME (AVERAGE OVER 1000 RUNS).

Scheme	ORRC				
(ℓ_1, ℓ_2)	$(2,2)$	$(4,4)$	$(7,7)$	$(20,20)$	$(30,30)$
Setup	0.15	0.30	0.54	1.54	2.25
Commit	1.84	3.56	6.36	18.21	28.45
VerCommit	1.66	3.19	5.72	16.56	25.37
Open	0.18	0.30	0.51	1.10	1.71
VerOpen	0.13	0.13	0.14	0.14	0.15
OpenLDP	0.65	0.75	0.85	1.14	1.50
VerOpenLDP	0.49	0.51	0.52	0.51	0.56
Gen	0.03	0.03	0.03	0.03	0.03
Sign	0.08	0.12	0.18	0.39	0.61
Ver	0.11	0.15	0.20	0.44	0.62

TABLE 4. SIZE IN BYTES OF THE ORRC SCHEME ELEMENTS

Scheme	ORRC				
(ℓ_1, ℓ_2)	$(2,2)$	$(4,4)$	$(7,7)$	$(20,20)$	$(30,30)$
set	480	864	1440	3936	5856
c	288	544	928	2592	3872
π_*	1664	3328	5824	16640	24960
π	96	96	96	96	96
$\hat{\pi}$	448	448	448	448	448

Comparison with other works. We compare the performance of ORRC with the alternative solutions that we considered relevant in the related works [11], [12].

We saw in the introduction that [11] seems to be adaptable for a scenario similar to ours, using the binomial mechanism on an integer value (not included in a predefined interval) instead of the randomised response. In this protocol, to apply a binomial noise, the opener must *flip* N *coin tosses* to generate random bits, and the time and size complexity of generating and verifying the proof are linear in N. Furthermore, N varies according to the level of privacy; it is proportional to $1/\epsilon^2$, so the more privacy we want, the smaller epsilon is, the larger N will be. The authors estimate the time for generating and verifying a proof to be equivalent to π_* and $\hat{\pi}$ in our setting for an epsilon $\epsilon = 0.095$, corresponding to $N = 262144$. They use the same elliptic curve and hardware similar to ours. For these parameters, generation takes 53 seconds and verification 43 seconds. In our case, for ℓ_2 set as $\ell_2 = 4$ (resp. $\ell_2 = 7$, $\ell_2 = 20$, and $\ell_2 = 30$) and $\epsilon = 0.095$, we have $\ell_1 = 8$ (resp. $\ell_1 = 11$, $\ell_1 = 24$, and $\ell_1 = 34$). We will therefore have performances similar to that presented in Table 3, *i.e.* a maximum of about thirty milliseconds for a set of one billion values, which is more efficient. The authors do not give the size of their proof in [11].

[12] uses the Groth-16 [14] SNARK for an application similar to ours. The efficiency of a SNARK depends on the number of gates in the evaluated circuit (for Groth-16 [14], the number of exponentiations grows linearly with the number of wires and multiplication gates in the proven arithmetic circuit). The size, on the other hand, is constant and very small (only a few group elements). Note however that, as already explained in the introduction, although the authors mention that their proof could be used on data signed in an anonymous credential resulting in a scheme similar to ours, they do not include this signature in the implemented circuit used for the efficiency evaluation. This would require the use of a more complex circuit and should therefore degrade the performance of this approach, which must be kept in mind when comparing it with our work. They use Circome [13] to implement the circuit corresponding to the randomized response mechanism and the exponential mechanism on a set of 128 values. Unfortunately, they do not specify the computational complexity required to generate the SNARK with their circuit, depending on the parameters used. They evaluate the efficiency using the exponential mechanism only. The execution time for generating/verifying the proof takes around 2.2 seconds in a setting comparable to ours with parameters $n_2 = 128$ and $\epsilon = 10$. For comparison, with such parameters, we would have $\ell_2 = 7$ and $\ell_1 = 1$, which would take less than 10 milliseconds with our protocol. On the other hand, our proofs are not of a constant size, but we believe that our analysis shows that the size is reasonable even for large parameters. Note also that even if the proofs are of fixed size in SNARKs, this is not the case for the setup. For instance, the SNARK evaluated in [12] requires to store a proving key of 3.4 megabytes and a public verification key of 3.5 kilobytes. Note also that this work requires strong hypotheses (pairings, generic group model, random oracle model), whereas we only need standard hypotheses and the random oracle model.

6. Conclusion

In this paper, we proposed a security model and a proven scheme for a new primitive called LDP commitment. Our scheme uses standard cryptographic tools and is very efficient. We have implemented it in Rust to analyse its performance and show that it can be used in practice. By signing an LDP commitment, a user allows a delegate to add noise before revealing their data to a recipient who will be convinced that they indeed received the data signed by the user with the correct LDP mechanism applied to it. This enables, for example, the protection of patient data used in a study to be guaranteed at the same time as the publication of the anonymized (signed) data to verify the statistical calculations made in the study. A natural extension of our work would be to instantiate our model with schemes allowing the use of other LDP mechanisms, such as the exponential mechanism. Another extension would be to find a way of adapting our primitive for use on non-discrete sets. Finally, a natural limitation of our approach is that the delegate (*i.e.* the hospital server) can reveal the non-noise values on an auxiliary channel if it colludes with the verifier. In future work, we could consider sharing the committed data among several servers so that it is not possible for the verifier to learn them without corrupting a threshold number of servers.

Acknowledgments

This work was supported by the France 2030 Projects PEPR Tracia with reference 22-PESN-0006, PEPR iPoP with reference 22-PECY-0002 and AMI CMA CyberINSA with reference 23-CMAS-0019.

References

[1] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *Advances in Cryptology - EUROCRYPT 2006*, S. Vaudenay, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 486–503.

[2] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. D. Smith, "What can we learn privately?" in *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*. IEEE Computer Society, 2008, pp. 531–540.

[3] W. Liu, Y. Zhang, H. Yang, and Q. Meng, "A survey on differential privacy for medical data analysis," *Annals of Data Science*, pp. 1–15, 2023.

[4] M. Adnan, S. Kalra, J. C. Cresswell, G. W. Taylor, and H. R. Tizhoosh, "Federated learning and differential privacy for medical image analysis," *Scientific Reports*, vol. 12, no. 1953, 2022.

[5] National Academies of Sciences, Engineering, and Medicine, "Reproducibility and replicability in science," *The National Academies Press*, 2019.

[6] J. Lee and C. Clifton, "How much is enough? choosing ϵ for differential privacy," in *Information Security, 14th International Conference, ISC 2011, Xi'an, China, October 26-29, 2011. Proceedings*, ser. Lecture Notes in Computer Science, X. Lai, J. Zhou, and H. Li, Eds., vol. 7001. Springer, 2011, pp. 325–340.

[7] C. Li, D. Y. Li, G. Miklau, and D. Suciu, "A theory of pricing private data," *ACM Trans. Database Syst.*, vol. 39, no. 4, pp. 34:1–34:28, 2014.

[8] A. Ambainis, M. Jakobsson, and H. Lipmaa, "Cryptographic randomized response techniques," in *Public Key Cryptography – PKC 2004*, F. Bao, R. Deng, and J. Zhou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 425–438.

[9] A. Cheu, A. Smith, and J. Ullman, "Manipulation attacks in local differential privacy," in *2021 IEEE Symposium on Security and Privacy (SP)*, 2021, pp. 883–900.

[10] F. Kato, Y. Cao, and M. Yoshikawa, "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*. Berlin, Heidelberg: Springer-Verlag, 2021, p. 43–60. [Online]. Available: https://doi.org/10.1007/978-3-030-81242-3_3

[11] A. Biswas and G. Cormode, "Interactive proofs for differentially private counting," in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security, CCS 2023, Copenhagen, Denmark, November 26-30, 2023*, W. Meng, C. D. Jensen, C. Cremers, and E. Kirda, Eds. ACM, 2023, pp. 1919–1933. [Online]. Available: https://doi.org/10.1145/3576915.3616681

[12] G. Munilla Garrido, J. Sedlmeir, and M. Babel, "Towards verifiable differentially-private polling," in *Proceedings of the 17th International Conference on Availability, Reliability and Security*, ser. ARES '22. New York, NY, USA: Association for Computing Machinery, 2022. [Online]. Available: https://doi.org/10.1145/3538969.3538992

[13] M. Bellés-Muñoz, M. Isabel, J. L. Muñoz-Tapia, A. Rubio, and J. Baylina, "Circom: A circuit description language for building zero-knowledge applications," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 6, pp. 4733–4751, 2023.

[14] J. Groth, "On the size of pairing-based non-interactive arguments," in *Advances in Cryptology – EUROCRYPT 2016*, M. Fischlin and J.-S. Coron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 305–326.

[15] A. Roy Chowdhury, C. Wang, X. He, A. Machanavajjhala, and S. Jha, "Cryptϵ: Crypto-Assisted Differential Privacy on Untrusted Servers," in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 603–619. [Online]. Available: https://doi.org/10.1145/3318464.3380596

[16] J. Böhler and F. Kerschbaum, "Secure multi-party computation of differentially private median," in *29th USENIX Security Symposium (USENIX Security 20)*. USENIX Association, Aug. 2020, pp. 2147–2164. [Online]. Available: https://www.usenix.org/conference/usenixsecurity20/presentation/boehler

[17] A. G. Sébert, M. Checri, O. Stan, R. Sirdey, and C. Gouy-Pailler, "Combining homomorphic encryption and differential privacy in federated learning," in *2023 20th Annual International Conference on Privacy, Security and Trust (PST)*, 2023, pp. 1–7.

[18] I. Mironov, O. Pandey, O. Reingold, and S. Vadhan, "Computational differential privacy," in *Advances in Cryptology - CRYPTO 2009*, S. Halevi, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 126–142.

[19] C. Brzuska, M. Fischlin, T. Freudenreich, A. Lehmann, M. Page, J. Schelbert, D. Schröder, and F. Volk, "Security of sanitizable signatures revisited," in *Public Key Cryptography–PKC 2009: 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings 12*. Springer, 2009, pp. 317–336.

[20] X. Bultel and P. Lafourcade, "Unlinkable and strongly accountable sanitizable signatures from verifiable ring signatures," in *Cryptology and Network Security: 16th International Conference, CANS 2017, Hong Kong, China, November 30—December 2, 2017, Revised Selected Papers*. Springer, 2018, pp. 203–226.

[21] S. Canard and A. Jambert, "On extended sanitizable signature schemes," in *Topics in Cryptology-CT-RSA 2010: The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1-5, 2010. Proceedings*. Springer, 2010, pp. 179–194.

[22] E. Boyle, S. Goldwasser, and I. Ivan, "Functional signatures and pseudorandom functions," in *Public-Key Cryptography – PKC 2014*, H. Krawczyk, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 501–519.

[23] M. Backes, S. Meiser, and D. Schröder, "Delegatable functional signatures," in *Public-Key Cryptography – PKC 2016*, C.-M. Cheng, K.-M. Chung, G. Persiano, and B.-Y. Yang, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 357–386.

[24] M. Bellare and G. Fuchsbauer, "Policy-based signatures," in *Public-Key Cryptography – PKC 2014*, H. Krawczyk, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 520–537.

[25] R. Johnson, D. A. Molnar, D. X. Song, and D. A. Wagner, "Homomorphic signature schemes," in *The Cryptographer's Track at RSA Conference*, 2002. [Online]. Available: https://api.semanticscholar.org/CorpusID:13965939

[26] M. Blum, P. Feldman, and S. Micali, "Non-Interactive Zero-Knowledge and Its Applications," in *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, ser. STOC '88. New York, NY, USA: Association for Computing Machinery, 1988, p. 103–112. [Online]. Available: https://doi.org/10.1145/62212.62222

[27] P. Kairouz, S. Oh, and P. Viswanath, "Extremal mechanisms for local differential privacy," *J. Mach. Learn. Res.*, vol. 17, pp. 17:1–17:51, 2016.

[28] V. Shoup, "Sequences of games: a tool for taming complexity in security proofs," Cryptology ePrint Archive, Paper 2004/332, 2004, https://eprint.iacr.org/2004/332. [Online]. Available: https://eprint.iacr.org/2004/332

[29] S. L. Warner, "Randomised response: a survey technique for eliminating evasive answer bias," *Journal of the American Statistical Association*, vol. 60, no. 309, pp. 63–69, 1965.

[30] D. Chaum and T. P. Pedersen, "Wallet Databases with Observers," in *Advances in Cryptology — CRYPTO' 92*. Springer, 1993.

[31] R. Cramer, I. Damgård, and B. Schoenmakers, "Proofs of partial knowledge and simplified design of witness hiding protocols," in *Advances in Cryptology—CRYPTO'94: 14th Annual International Cryptology Conference Santa Barbara, California, USA August 21–25, 1994 Proceedings*. Springer, 2001, pp. 174–187.

[32] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology—CRYPTO'91: Proceedings*. Springer, 2001, pp. 129–140.

[33] M. Fischlin and A. Mittelbach, “An overview of the hybrid argument,” Cryptology ePrint Archive, Paper 2021/088, 2021, https://eprint.iacr.org/2021/088. [Online]. Available: https://eprint.iacr.org/2021/088

[34] A. Fiat and A. Shamir, “How To Prove Yourself: Practical Solutions to Identification and Signature Problems,” in *CRYPTO’ 86*. Springer, 1987.

[35] J. Camenisch and V. Shoup, “Practical Verifiable Encryption and Decryption of Discrete Logarithms,” in *Advances in Cryptology - CRYPTO 2003*, D. Boneh, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 126–144.

[36] C. P. Schnorr, “Efficient identification and signatures for smart cards,” in *Advances in Cryptology — CRYPTO’ 89 Proceedings*, G. Brassard, Ed. New York, NY: Springer New York, 1990, pp. 239–252.

[37] “Cryptographic Commitments on Anonymizable Data (Full Version and Implementation).” [Online]. Available: https://github.com/charlene-j/Cryptographic-Commitments-on-Anonymizable-Data

[38] I. A. Lovecruft and H. de Valence, “curve25519_dalek.” [Online]. Available: https://docs.rs/curve25519-dalek/latest/curve25519_dalek/

[39] X. Bultel and C. Olivier-Anclin, “Taming delegations in anonymous signatures: k-times anonymity for proxy and sanitizable signature,” in *Cryptology and Network Security*, M. Kohlweiss, R. Di Pietro, and A. Beresford, Eds. Singapore: Springer Nature Singapore, 2025, pp. 165–186.

[40] E. Bangerter, T. Briner, W. Henecka, S. Krenn, A.-R. Sadeghi, and T. Schneider, “Automatic generation of sigma-protocols,” in *Public Key Infrastructures, Services and Applications*, F. Martinelli and B. Preneel, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 67–82.

[41] A. Shamir, “How to share a secret,” *Commun. ACM*, vol. 22, no. 11, p. 612–613, Nov. 1979. [Online]. Available: https://doi.org/10.1145/359168.359176

Appendix A. Proof of Theorem 1

Proof. By abuse of notation, we simply note $\Pr[b = b_*]$ the probability:

$$\Pr\left[\begin{array}{l} b \xleftarrow{\$} \{0,1\}; \hat{m} \leftarrow \mathsf{LDP}(m_b); \\ b_* \leftarrow \mathsf{OptiGuess}(\lambda, \mathsf{LDP}, \hat{m}, m_0, m_1) \end{array} : b = b_*\right].$$

When b is set to a value $x \in \{0,1\}$ (resp. $\hat{m}$ to a value $\hat{x} \in \mathcal{M}$), we will note $\mathsf{Pr}[b = b_* | b = x]$ (resp. $\mathsf{Pr}[b = b_* | \hat{m} = \hat{x}]$). Since $\Pr[b = 0] = \Pr[b = 1] = \frac{1}{2}$, we have:

$$\begin{aligned}\Pr[b = b_*] &= \Pr[b = 0] \cdot \Pr[b = b_* | b = 0] + \\ &\quad \Pr[b = 1] \cdot \Pr[b = b_* | b = 1] \\ &= \frac{1}{2}\left(\Pr[b = b_* | b = 0] + \Pr[b = b_* | b = 1]\right).\end{aligned}$$

We first evaluate the first probability:

$$\begin{aligned}&\Pr[b = b_* | b = 0] \\ &= \sum_{\hat{x} \in \mathcal{M}} \Pr[\hat{m} = \hat{x} | b = 0] \cdot \Pr[b = b_* | b = 0 \land \hat{m} = \hat{x}] \\ &= \sum_{\hat{x} \in \mathcal{M}} \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)] \cdot \Pr[b = b_* | b = 0 \land \hat{m} = \hat{x}]\end{aligned}$$

For all comparison operators $\square \in \{=, <, >\}$, we define the set $\mathcal{M}_\square$ as follows:

$$\hat{x} \in \mathcal{M}_\square \Leftrightarrow \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)] \,\square\, \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_1)].$$

Let $\mathcal{P} = \{\mathcal{M}_=, \mathcal{M}_<, \mathcal{M}_>\}$ be a set, we have that $\mathcal{P}$ is a partition of $\mathcal{M}$. We recall that by definition, $\mathsf{OptiGuess}$ returns 0 with probability:

- 1 if $\Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)] > \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_1)]$,
- $\frac{1}{2}$ if $\Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)] = \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_1)]$, and
- 0 if $\Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)] < \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_1)]$.

We deduce that:

$$\begin{aligned}&\Pr[b = b_* | b = 0] \\ &= \sum_{\hat{x} \in \mathcal{M}} \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)] \cdot \Pr[b = b_* | b = 0 \land \hat{m} = \hat{x}] \\ &= \sum_{S \in \mathcal{P}} \sum_{\hat{x} \in S} \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)] \cdot \Pr\left[b = b_* \middle| \begin{array}{c} b = 0 \land \\ \hat{m} = \hat{x} \end{array}\right] \\ &= \sum_{\hat{x} \in \mathcal{M}_>} \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)] + \\ &\quad \frac{1}{2} \cdot \sum_{\hat{x} \in \mathcal{M}_=} \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)] + \\ &\quad 0 \cdot \sum_{\hat{x} \in \mathcal{M}_<} \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)] \\ &= \sum_{\hat{x} \in \mathcal{M}_>} \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)] + \\ &\quad \frac{1}{2} \sum_{\hat{x} \in \mathcal{M}_=} \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)].\end{aligned}$$

In a similar way, we have:

$$\begin{aligned}\Pr[b = b_* | b = 1] = &\sum_{\hat{x} \in \mathcal{M}_<} \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_1)] + \\ &\frac{1}{2} \sum_{\hat{x} \in \mathcal{M}_=} \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_1)].\end{aligned}$$

Furthermore:

- for all $\hat{x} \in \mathcal{M}_>$,
$$\begin{aligned}&\max(\Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)], \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_1)]) \\ &= \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)].\end{aligned}$$
- for all $\hat{x} \in \mathcal{M}_=$,
$$\begin{aligned}&\max(\Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)], \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_1)]) \\ &= \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)] = \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_1)].\end{aligned}$$
- for all $\hat{x} \in \mathcal{M}_<$,
$$\begin{aligned}&\max(\Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)], \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_1)]) \\ &= \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_1)].\end{aligned}$$

We deduce:

$$\begin{aligned}&\Pr[b = b_* | b = 0] + \Pr[b = b_* | b = 1] \\ &= \sum_{\hat{x} \in \mathcal{M}_>} \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)] + \\ &\quad \frac{1}{2} \sum_{\hat{x} \in \mathcal{M}_=} \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)] + \\ &\quad \sum_{\hat{x} \in \mathcal{M}_<} \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_1)] + \\ &\quad \frac{1}{2} \sum_{\hat{x} \in \mathcal{M}_=} \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_1)] \\ &= \sum_{\hat{x} \in \mathcal{M}} \max(\Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)], \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_1)]).\end{aligned}$$

This leads to the result of the theorem:

$$\begin{aligned}&\Pr[b = b_*]\\&= \frac{1}{2}\sum_{\hat{x}\in\mathcal{M}} \max(\Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)], \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_1)]).\end{aligned}$$

This concludes the proof.

$\square$

Appendix B. Extensions

In this section, we propose three extensions to our scheme. The first concerns the case where the same commitment can be opened several times using the LDP mechanism with different seeds. The second shows that it is possible to open the commitment with a LDP parameter ϵ larger (*i.e.* less privacy, more utility) than the one chosen at the time of the commitment without interacting with the user having committed the data. Finally, the last extension shows how to adapt our scheme to choose a parameter n_1 that is not a power of 2.

Extension for Multiple Openings. In our application, one can imagine cases where the same data is used by independent researchers (who do not necessarily trust each other) in several different studies. In this case, it should be possible to open the commitment several times with different random seeds $\hat{\theta}$. However, our scheme supports only one opening using LDP (or several openings to non-colluding users). For instance, given two openings for the commitment of m giving the messages $\hat{m}$ and $\hat{m}'$ for the seeds $\hat{\theta} = (\hat{s}, \hat{t})$ and $\hat{\theta}' = (\hat{s}', \hat{t}')$ such that $\hat{\theta} \neq \hat{\theta}'$, if $\hat{m} = \hat{m}'$, we can deduce that $\hat{m} = \hat{m}' = m$, otherwise, if $\hat{m} \oplus \hat{m}' = \hat{t} \oplus \hat{t}'$, we can deduce with a high probability that $\hat{m} \neq m$ and $\hat{m}' \neq m$. Note that, although it may be possible to limit the leakage of information about the message, the problem is inherent in the primitive: an attacker that could try each of the seeds will always find the original message with certainty. The number of openings with LDP cannot therefore be variable and must be fixed at the time of signing.

We propose here a simple solution to allow a fixed number N of openings, at the cost of a linear factor in N on the size of the commitment and its generation time (note that it is in any case undesirable to open the same data several times with LDP because of the composition theorem [1] which shows that opening N times the same data with an ϵ-differentially private mechanism amounts to applying a $(N\epsilon)$-differentially private mechanism).

The solution is simply to commit to the same message N times with ORRC, and to open a different commitment for each opening query. We denote $\mathsf{c} = (\mathsf{c}_i)_{i\in[\![N]\!]}$ the commitment constructed in this way, with for each $i \in [\![N]\!]$, $\mathsf{c}_i = (y_i, (A_{(i,j,1)})_{j\in[\![\ell_1]\!]}, (A_{(i,j,2)}, B_{(i,j,0)}, B_{(i,j,1)})_{j\in[\![\ell_2]\!]})$, and $\mathsf{k} = (\mathsf{k}_i)_{i\in[\![N]\!]}$ the corresponding opening secret key. In the case where the user having committed to the message is not considered to be honest, it is also necessary to ensure that it is indeed the same message that has been committed in each of the N ORRC commitments. This can easily be shown using the following proofs of knowledge of a discrete logarithm for each (i, j) in $([\![N]\!]\backslash\{1\}) \times j \in [\![\ell_2]\!]$ where $\mathsf{k}_i^* = \mathsf{k}_i/\mathsf{k}_1$:

$$\pi_{(i,j)} \leftarrow \mathsf{NIP}\left\{\mathsf{k}_i^* : y_1^{\mathsf{k}_i^*} = y_i \wedge A_{(1,j,2)}^{\mathsf{k}_i^*} = A_{(i,j,2)}\right\}.$$

Combined with the other proofs generated during the commitment, this proof proves that $y_1 = g^{\mathsf{k}_1}$ and $A_{(1,j,2)} = f_{(j,0)}^{\mathsf{k}_1}$ and $y_i = g^{\mathsf{k}_i}$ and $f_{(j,0)}^{\mathsf{k}_i}$, or $y_1 = g^{\mathsf{k}_1}$ and $A_{(1,j,2)} = f_{(j,1)}^{\mathsf{k}_1}$ and $y_i = g^{\mathsf{k}_i}$ and $f_{(j,1)}^{\mathsf{k}_i}$, which proves that the j-th bit of the committed message is the same in the first and i-th commitments. Combining all these proofs, we have that all the committed messages are actually the same. We leave as an open problem the question of finding a way to allow multiple openings without a linear factor in time and size complexity for the generation of the commitment.

Opening with Larger ϵ. In the context of our applications, where the user who commits the data and the user who reveals it are two different entities, it may be useful to decide at the time of opening which ϵ will be used. This would make it possible to decide on the ϵ according to the intended use of the data, which is not necessarily known in advance. This would be particularly useful for our secondary application of selling private data, since the price is set according to the added noise, so this extension makes it possible to negotiate a price/privacy compromise with the buyer.We note that this is possible with our scheme by reducing the parameter $n_1 = 2^{\ell_1}$. Remember that the commitment uses a binary string s of ℓ_1 bits, so that each of its bits is committed to the $(A_{i,1})_{i\in[\![\ell_1]\!]}$, and which is used to decide whether the opening with the LDP will return the value actually committed or a randomly chosen value (depending on whether $s = \hat{s}$ or not, where $\hat{s}$ is also a string of ℓ_1 bits chosen by the verifier). If we need to use an LDP mechanism parameterized by $n_1' = 2^{\ell_1'}$ with $l_1' < \ell_1$, we simply choose a $\hat{s}$ of ℓ_1' bits and compare it with the binary string s' made up of the first ℓ_1' bits of s, using the elements $(A_{i,1})_{i\in[\![\ell_1']\!]}$.

Fine-grained choice of parameter n_1. As it stands, our scheme requires n_1 to be a power of 2, which does not allow us to precisely choose the probability $1/n_1$ for the opener to open on the original value. To do this, we should set s to be the ℓ_1-bit coding of a number chosen at random between 0 and $n_1 - 1$ where $n_1 < 2^{\ell_1}$, and ask the verifier that $\hat{s}$ also be the ℓ_1-bit coding of a number less than n_1. On the other hand, it would be necessary to prove that the committed s has been chosen correctly, *i.e.* that $s < n_1$. Such a proof is constructed in [39] for another application. Note that this proof concerns commitments constructed in the same way as in our case and can be used without adaptation (for each bit, the two possible values correspond to two group generators, and the generator corresponding to the actual bit value is used to compute the exponentiation of a secret), and that the complexity in time and size of this proof is in $O(\ell_1)$, so its use does not affect the overall complexity in $O(\max(\log_2(n_1), \log_2(n_2))$ of our construction.

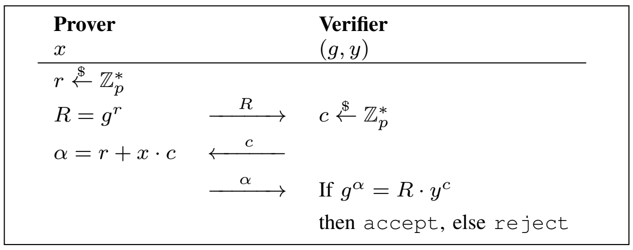

Figure 4. Discrete logarithms equality proof

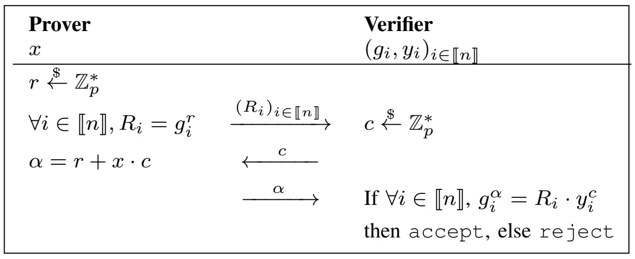
Prover
x
Verifier
$(g_i, y_i)_{i \in [\![n]\!]}$
$r \xleftarrow{\$} \mathbb{Z}_p^*$
$\forall i \in [\![n]\!], R_i = g_i^r$
$\xrightarrow{(R_i)_{i \in [\![n]\!]}}$
$c \xleftarrow{\$} \mathbb{Z}_p^*$
$\alpha = r + x \cdot c$
$\xleftarrow{c}$
$\xrightarrow{\alpha}$
If $\forall i \in [\![n]\!]$, $g_i^\alpha = R_i \cdot y_i^c$
then accept, else reject

Figure 5. Discrete logarithms equality proof

Appendix C. ZKP Instantiation

In this section, we describe how to instantiate the various zero-knowledge proofs to implement our protocols.

C.1. Building blocks

We first recall the Schnorr protocol [36], which is an interactive proof allowing a prover to prove to a verifier the knowledge of the discrete logarithm of an element y in base g in a group $\mathbb{G}$ of prime order p. This protocol is given in Fig. 4.

The second building block we need is a proof that elements have the same discrete logarithm, i.e. for n pairs of elements $(g_i, y_i)_i \in [\![n]\!]$ of a group $\mathbb{G}$ of prime order p, there exists a (secret) x such that $g_i^x = y_i$ for all i. Such an (interactive) proof is given in [30], and is shown in Fig. 5.

Our second building block is a proof that two elements have different discrete logarithms, i.e. for the elements (g_1, y_1, g_2, y_2), there exists a (secret) x such that $g_1^x = y_1$ and $g_2^x \neq y_2$. Such an (interactive) proof is given in [35], and is built from a proof that for the elements $(g_1, h_1, y_1, g_2, h_2, y_2)$, there exists two (secrets) x and w such that $g_1^x \cdot h_1^w = y_1$ and $g_2^x \cdot h_2^w = y_2$. This proof is instantiated with the protocol given in Fig. 6 with $n = 2$.

Then, the proof of knowledge of x such that $g_1^x = y_1$ and $g_2^x \neq y_2$ given in [30] is formulated in Fig. 7. The idea behind this protocol is to prove, with the proof of Fig. 6, knowledge of two elements w‘ and x' such that $g_1^{w‘}/y_1^{x'} = 1$ and $g_1^{w‘}/y_1^{x'} = y$‘ for a y' which verifies $y' \neq 1$. Thus, by setting $x = w‘/x'$, we have indeed proven the knowledge of an x such that $g_1^x = y_1$ and $g_2^x \neq y_2$. For our work, we actually need to prove a slightly different relation, where two elements y_1 and y_2 have the same discrete logarithm in bases g_1 and g_2, but a third element y_3 has a different discrete logarithm in base g_3, i.e. we know an x such that $g_1^x = y_1$ and $g_2^x = y_2$ but $g_3^x \neq y_3^x$.

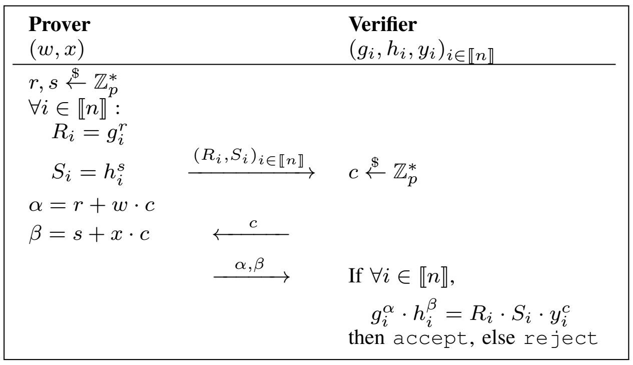
Prover
(w, x)
Verifier
$(g_i, h_i, y_i)_{i \in [\![n]\!]}$
$r, s \xleftarrow{\$} \mathbb{Z}_p^*$
$\forall i \in [\![n]\!]$:
$R_i = g_i^r$
$S_i = h_i^s$
$\xrightarrow{(R_i, S_i)_{i \in [\![n]\!]}}$
$c \xleftarrow{\$} \mathbb{Z}_p^*$
$\alpha = r + w \cdot c$
$\beta = s + x \cdot c$
$\xleftarrow{c}$
$\xrightarrow{\alpha, \beta}$
If $\forall i \in [\![n]\!]$,
$g_i^\alpha \cdot h_i^\beta = R_i \cdot S_i \cdot y_i^c$
then accept, else reject

Figure 6. Building block for discrete logarithms inequality proof

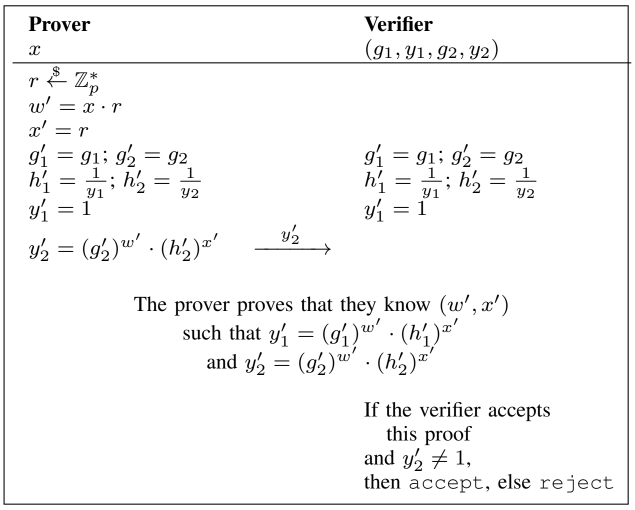
Prover
x
Verifier
(g_1, y_1, g_2, y_2)
$r \xleftarrow{\$} \mathbb{Z}_p^*$
$w' = x \cdot r$
$x' = r$
$g_1' = g_1$; $g_2' = g_2$
$h_1' = \frac{1}{y_1}$; $h_2' = \frac{1}{y_2}$
$y_1' = 1$
$g_1' = g_1$; $g_2' = g_2$
$h_1' = \frac{1}{y_1}$; $h_2' = \frac{1}{y_2}$
$y_1' = 1$
$y_2' = (g_2')^{w'} \cdot (h_2')^{x'}$
$\xrightarrow{y_2'}$
The prover proves that they know (w', x')
such that $y_1' = (g_1')^{w'} \cdot (h_1')^{x'}$
and $y_2' = (g_2')^{w'} \cdot (h_2')^{x'}$
If the verifier accepts
this proof
and $y_2' \neq 1$,
then accept, else reject

Figure 7. Discrete logarithms inequality proof

This proof, given in Fig. 8, follows naturally from the one given in Fig. 7 using the protocol given in Fig. 6 with $n = 3$.

These proofs are Schnorr-like sigma protocols [40], which means that they follow the structure given in Fig. 9, where $(x, y) \in \mathcal{R}$ for a given relation $\mathcal{R}$, and where Com, Resp, and Ver are polynomial-time algorithms in $|p|$.

For instance, for the Schnorr protocol (Fig. 9), $\mathsf{Com}(y, r) = g^r$, $\mathsf{Resp}(y, r, c) = r + x \cdot c$, and $\mathsf{Ver}(y, R, c, \alpha) = (g^\alpha = R \cdot y^c)$. Note that a Schnorr-like sigma protocol can be turned into a non-interactive proof by using a hash function $H : \{0, 1\}^* \to \mathbb{Z}_p^*$ modelled by a random oracle [34]. To remove the interaction, the prover can simply choose the challenge $c = H(y, R)$ instead of asking the verifier.

To prove knowledge of the secret (x_1, x_2) such that $(y_1, x_1) \in \mathcal{R}_1 \wedge (y_2, x_2) \in \mathcal{R}_2$ for some instances and relations y_1, $\mathcal{R}_1$, y_2, and $\mathcal{R}_2$, it is sufficient to prove $(y_1, x_1) \in \mathcal{R}_1$ and $(y_2, x_2) \in \mathcal{R}_2$ using two independent proofs. Recall that the structure of a Schnorr proof simulator is always the same: the simulator $\mathsf{Sim}(y)$ picks at random c and α, builds R from (y, c, α), and returns (R, c, α). For instance, for the Schnorr protocol (Fig. 9), $\mathsf{Sim}(y)$ picks α and c and returns $(g^\alpha / y^c, c, \alpha)$. Thus, to prove several independent relations at the same time, the same challenge can be used without compromising the zero-knowledge propery. The protocol in Fig. 10 proves that $\bigwedge_{i=1}^{n} (y_i, x_i) \in \mathcal{R}_i$ for n statements y_i of n relations

Prover		**Verifier**
x		$(g_1, y_1, g_2, y_2, g_3, y_3)$
$r \xleftarrow{\$} \mathbb{Z}_p^*$		
$w' = x \cdot r$		
$x' = r$		
$\forall i \in [\![3]\!]:$		$\forall i \in [\![3]\!]:$
$g'_i = g_i$		$g'_i = g_i$
$h'_i = \frac{1}{y_i}$		$h'_i = \frac{1}{y_i}$
$y'_1 = 1;\ y'_2 = 1$		$y'_1 = 1;\ y'_2 = 1$
$y'_3 = (g'_3)^{w'} \cdot (h'_3)^{x'}$	$\xrightarrow{y'_3}$	
	The prover proves that they know (w', x') such that $y'_1 = (g'_1)^{w'} \cdot (h'_1)^{x'}$ and $y'_2 = (g'_2)^{w'} \cdot (h'_2)^{x'}$ and $y'_3 = (g'_3)^{w'} \cdot (h'_3)^{x'}$	
		If the verifier accepts this proof and $y'_3 \neq 1$, then `accept`, else `reject`

Figure 8. Discrete logarithms inequality proof for 3 elements

Prover		**Verifier**
x		y
Picks r at random		
$R = \mathsf{Com}(y, r)$	$\xrightarrow{R}$	$c \xleftarrow{\$} \mathbb{Z}_p^*$
$\alpha = \mathsf{Resp}(y, r, c)$	$\xleftarrow{c}$	
	$\xrightarrow{\alpha}$	Return $\mathsf{Ver}(y, R, c, \alpha)$

Figure 9. Schnorr-like sigma protocol

$\mathcal{R}_i$ knowing n secret witnesses x_i, using the Schnorr-like sigma protocols $(\mathsf{Com}_i, \mathsf{Resp}_i, \mathsf{Ver}_i)$. Note that this is a Schnorr-like sigma protocol.

In [31], from n sigma protocols $(\mathsf{Com}_i, \mathsf{Resp}_i, \mathsf{Ver}_i)$ associated with n relations $\mathcal{R}_i$, Cramer *et. al.* show how to construct a sigma protocol that allows a prover to prove knowledge of t secrets x_i and t instances y_i among n (without revealing which ones) such that for each of these secrets, $(y_i, x_i) \in \mathcal{R}_i$. Their construction uses a threshold secret sharing, which we will instantiate using Shamir's secret sharing [41]. The latter uses Lagrange interpolation in $\mathbb{Z}_p^*$:

$$L_{(a_i,b_i)_{i\in[\![n]\!]}}(x) = \sum_{j=1}^{n} b_j \prod_{i=1;i\neq j}^{n} \frac{x - a_i}{a_j - a_i}.$$

This construction is shown in Fig. 11, note that this is a Schnorr-like sigma protocol.

Prover		**Verifier**
$(x_i)_{i\in[\![n]\!]}$		$(y_i)_{i\in[\![n]\!]}$
Picks $(r_i)_{i\in[\![n]\!]}$ at random		
$\forall i \in [\![n]\!], R_i = \mathsf{Com}_i(y_i, r_i)$	$\xrightarrow{(R_i)_{i\in[\![n]\!]}}$	$c \xleftarrow{\$} \mathbb{Z}_p^*$
$\forall i \in [\![n]\!], \alpha_i = \mathsf{Resp}_i(y_i, r_i, c)$	$\xleftarrow{c}$	
	$\xrightarrow{(\alpha_i)_{i\in[\![n]\!]}}$	Return : $\bigwedge_{i=1}^{n} \mathsf{Ver}_i(y_i, R_i, c, \alpha_i)$

Figure 10. AND-proof for the relation $\bigwedge_{i=1}^{n}(y_i, x_i) \in \mathcal{R}_i$.

Prover		**Verifier**
$(x_i)_{i\in I}$ where $I \subseteq [\![n]\!]$		$(y_i)_{i\in[\![n]\!]}$
$\forall i \in [\![n]\!]$:		
If $i \in I$:		
picks r_i at random		
$R_i = \mathsf{Com}_i(y_i, r_i)$		
else:		
$(R_i, c_i, \alpha_i) \leftarrow \mathsf{Sim}_i(y_i)$	$\xrightarrow{(R_i)_{i\in[\![n]\!]}}$	$c_0 \xleftarrow{\$} \mathbb{Z}_p^*$
$\bar{I} = ([\![n]\!] \cup \{0\}) \backslash I$	$\xleftarrow{c_0}$	
$f(x) = L_{(i,c_i)_{i\in\bar{I}}}(x)$		
$\forall i \in I$:		
$\alpha_i = \mathsf{Resp}_i(y_i, r_i, f(i))$	$\xrightarrow{f,(\alpha_i)_{i\in[\![n]\!]}}$	If $f(0) = c_0$ and $\deg(f) = n - t$ and $\bigwedge_{i=1}^{n} \mathsf{Ver}_i(y_i, R_i, f(i), \alpha_i)$ then `accept`, else `reject`

Figure 11. Proof of partial knowledge.

Prover		**Verifier**
x_j		$(y_i)_{i\in[\![n]\!]}$
picks r_j at random		
$R_j = \mathsf{Com}_j(y_j, r_j)$		
$\forall i \in [\![n]\!] \backslash \{j\}$:		
$(R_i, c_i, \alpha_i) \leftarrow \mathsf{Sim}_i(y_i)$	$\xrightarrow{(R_i)_{i\in[\![n]\!]}}$	$c_0 \xleftarrow{\$} \mathbb{Z}_p^*$
$c_j = \frac{c_0}{\prod_{i=1;i\neq j}^{n} c_i}$	$\xleftarrow{c_0}$	
$\alpha_j = \mathsf{Resp}_j(y_j, r_j, c_j)$	$\xrightarrow{(c_i,\alpha_i)_{i\in[\![n]\!]}}$	If $c_0 = \prod_{i=1}^{n} c_i$ and $\bigwedge_{i=1}^{n} \mathsf{Ver}_i(y_i, R_i, c, \alpha_i)$ then `accept`, else `reject`

Figure 12. OR-proof for the relation $\bigvee_{i=1}^{n}(y_i, x_j) \in \mathcal{R}_i$.

In the particular case where we prove knowledge of only one secret among n, a threshold-free secret sharing is sufficient, so we can use multiplication in $\mathbb{Z}_p^*$ to share the challenge c. The resulting protocol, given in Fig. 12, allows a prover to prove knowledge of one x_j such that $\bigvee_{i=1}^{n}(y_i, x_j) \in \mathcal{R}_i$ more efficiently. Note that this is a Schnorr-like sigma protocol.

We will now evaluate the complexity in time and size of the non-interactive version of these proofs. For basic proofs (Fig. 4, 5, 6, and 7), the time is evaluated in the number of exponentiations in the group, and the size in the number of group elements. For constructions using other generic proofs (Fig. 9, 10, 11, and 12), the time is evaluated in number of executions of the least efficient generic proof, and the size in number of proofs for the proof system generating the largest proofs. Our complexity analysis is given in Fig. 13.

C.2. Instantiation of the proofs in our naive solution

This protocol uses a commitment scheme, which can be instantiated with the Pedersen commitment [32]. As a reminder, this scheme uses a group of prime order p and two public generators g and h. To commit a value m with a key k, the committer computes the commitment $c = g^k \cdot h^m$. To prove *a posteriori* that the commitment contains m, it suffices to prove knowledge of the key k such that $g^k = \left(\frac{c}{h^m}\right)$. This can be done with the Schnorr protocol (Fig. 4).

proof	Proof size	Prover time	Verifier time
Fig. 4	$O(1)$	$O(1)$	$O(1)$
Fig. 5	$O(n)$	$O(n)$	$O(n)$
Fig. 6	$O(n)$	$O(n)$	$O(n)$
Fig. 7	$O(1)$	$O(1)$	$O(1)$
Fig. 8	$O(1)$	$O(1)$	$O(1)$
Fig. 9	$O(1)$	$O(1)$	$O(1)$
Fig. 10	$O(n)$	$O(n)$	$O(n)$
Fig. 11	$O(n)$	$O(n)$	$O(n)$
Fig. 12	$O(n)$	$O(n)$	$O(n)$

Figure 13. Complexity in time and size of our building block proofs

C.2.1. Proof π_* from Commit. π_* proves that:

- Each possible m is committed at least $n_1 - 1$ times in c. We have already seen how to prove that a commitment contains a given message; it suffices to apply the transformation given in Fig. 11 to show prove at least $n_1 - 1$ commitments contain a given message, and repeat the proof for each possible message.
- The value committed in c_* is committed at least $n_1 + n_2 - 1$ times in c. The first part of the proof implies that if the message in c_* is committed $n_1 + n_2 - 1$ in c, then it is indeed one of the possible messages. To show that two commitments $c_* = g^{k_*} \cdot h^m$ and $c = g^k \cdot h^m$ contain the same message m, the prover must show that they know the keys k and k' such that $\left(\frac{c_*}{c}\right) = \left(\frac{g^{k_*}}{g^k}\right)$, which is equivalent to prove the knowledge of $(k_* - k)$ such that $\left(\frac{c_*}{c}\right) = g^{(k_*-k)}$. This can be done with the Schnorr protocol. By applying the transformation given in Fig. 11, we can prove that the message in c_* is committed at least $n_1 + n_2 - 1$ in c.

These proofs ensure that the value committed in c_* appears exactly $n_1 + n_2 - 1$ times in the commitments, and each of the other possible messages appear exactly $n_1 - 1$ times.

C.2.2. Proof π from Open. This is a proof that c_* contains a given message m. We have already shown that this can be done with the Schnorr protocol (Fig. 4).

C.2.3. Proof $\hat{\pi}$ from OpenLDP. This is a proof that one c_i contains a given message m. We have already shown that this can be done with the Schnorr protocol (Fig. 4).

C.3. Instantiation of the proofs in ORRC

C.3.1. Proof π_* from Commit. π_* is divided into three parts:

- π_{A_1} proves the following relation knowing x:

$$\bigwedge_{i=1}^{\ell_1} \left(\begin{array}{c} y = g^x \\ \wedge \left(\bigvee_{j=0}^{1} A_{(i,1)} = g_{i,j}^x \right) \end{array} \right),$$

which is equivalent to the relation:

$$\bigwedge_{i=1}^{\ell_1} \left(\begin{array}{l} (y = g^x \wedge A_{(i,1)} = g_{i,0}^x) \\ \vee (y = g^x \wedge A_{(i,1)} = g_{i,1}^x) \end{array} \right)$$

The part $(y = g^x \wedge A_{(i,1)} = g_{i,j}^x)$ for any i and j can be proven using the protocol given in Fig. 5. We can apply the transformation given in Fig. 12 to prove $(y = g^x \wedge A_{(i,1)} = g_{i,0}^x) \vee (y = g^x \wedge A_{(i,1)} = g_{i,1}^x)$. Finally, we apply the transformation given in Fig. 10 on the result to prove the full relation.
- π_{A_2} can be done in the same way.
- π_B proves the following relation knowing x:

$$\bigwedge_{i=1}^{\ell_2} \left(\begin{array}{c} y = g^x \\ \wedge \left(\begin{array}{l} \bigwedge_{j=0}^{1} B_{(i,j)} = h_{i,j}^x \\ \vee \bigwedge_{j=0}^{1} B_{(i,j)} = h_{i,1-j}^x \end{array} \right) \end{array} \right)$$

which is equivalent to the relation:

$$\bigwedge_{i=1}^{\ell_2} \left(\begin{array}{l} (y = g^x \wedge B_{(i,0)} = h_{i,0}^x \wedge B_{(i,1)} = h_{i,1}^x) \\ \vee (y = g^x \wedge B_{(i,0)} = h_{i,1}^x \wedge B_{(i,1)} = h_{i,0}^x) \end{array} \right)$$

The parts $(y = g^x \wedge B_{(i,0)} = h_{i,0}^x \wedge B_{(i,1)} = h_{i,1}^x)$ and $(y = g^x \wedge B_{(i,0)} = h_{i,1}^x \wedge B_{(i,1)} = h_{i,0}^x)$ for any i can be proven using the protocol given in Fig. 5. We can apply the transformation given in Fig. 12 to prove $((y = g^x \wedge B_{(i,0)} = h_{i,0}^x \wedge B_{(i,1)} = h_{i,1}^x) \vee (y = g^x \wedge B_{(i,0)} = h_{i,1}^x \wedge B_{(i,1)} = h_{i,0}^x)$. Finally, we apply the transformation given in Fig. 10 on the result to prove the full relation.

C.3.2. Proof π from Open. This is a proof of discrete logarithms equality, which can be instantiated with the protocol given in Fig. 5.

C.3.3. Proof $\hat{\pi}$ from OpenLDP. $\hat{\pi}$ proves the following relation knowing x:

$$\begin{array}{l} (y = g^x \wedge A_1 = \alpha_1^x \wedge A_2 = \alpha_2^x) \\ \vee (y = g^x \wedge A_1 \neq \alpha_1^x \wedge \beta = \gamma^x) . \end{array}$$

The relations $(y = g^x \wedge A_1 = \alpha_1^x \wedge A_2 = \alpha_2^x)$ can be proven using the protocol given in Fig. 5, and the relation $(y = g^x \wedge A_1 \neq \alpha_1^x \wedge \beta = \gamma^x)$ can be proven using the protocol given in Fig. 8. By applying the transformation given in Fig. 12 on these two proofs, we obtain a proof for the full relation.

C.4. Complexity evaluation

In Fig. 14, we evaluate the complexity in time and size of the proofs used in our naive protocol and in ORRC

proof	Protocol	Proof size	Prover/Verifier time
π_*	naive	$O(2^{\ell_1+\ell_2})$	$O(2^{\ell_1+\ell_2})$
	ORRC	$O(\max(\ell_1, \ell_2))$	$O(\max(\ell_1, \ell_2))$
π	naive	$O(1)$	$O(1)$
	ORRC	$O(1)$	$O(1)$
$\hat{\pi}$	naive	$O(1)$	$O(1)$
	ORRC	$O(1)$	$O(1)$

Figure 14. Complexity in time and size of the proofs used in our naive protocol and in ORRC with parameters $n_1 = 2^{\ell_1}$ and $n_2 = 2^{\ell_2}$

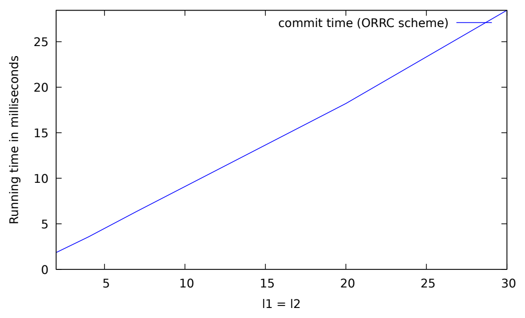

Figure 15. Computation time for Commit in ORRC as a function of ℓ_1 (with $\ell_2 = \ell_1$).

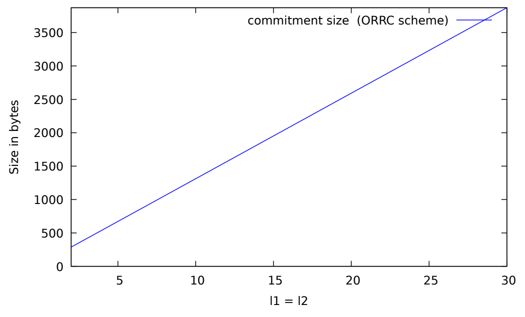

Figure 16. Commitment size in ORRC as a function of ℓ_1 (with $\ell_2 = \ell_1$).

with parameters $n_1 = 2^{\ell_1}$ and $n_2 = 2^{\ell_2}$. The time is evaluated in the number of exponentiations in the group, and the size in the number of group elements.

Appendix D. Graphical representation of the commitment cost as a function of (ℓ_1, ℓ_2) in ORRC.

In this section, we give a graphical representation of the results shown in Table 3 and Table 4 for the evolution over time of the execution of Commit (Fig. 15), the size of the commitment (Fig. 16), and the size of the proof π_* (Fig. 17) as a function of the parameters (ℓ_1, ℓ_2). As expected in our complexity analysis, these quantities are linear in (ℓ_1, ℓ_2) (and therefore logarithmic in $n_1 = 2^{\ell_1}$ and $n_2 = 2^{\ell_2}$).

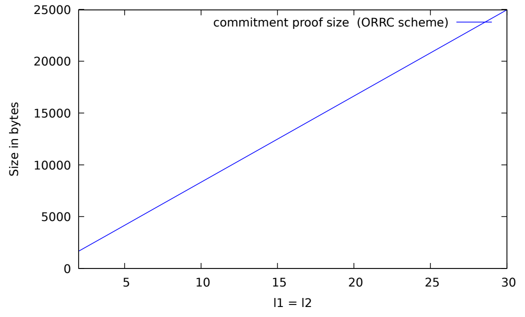

Figure 17. Proof π_* size in ORRC as a function of ℓ_1 (with $\ell_2 = \ell_1$).

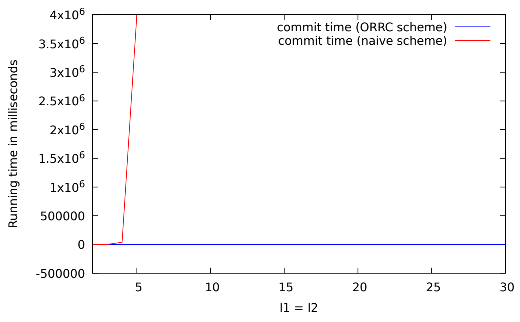

Figure 18. Comparison of the computation time for generating the commitment in the naive solution and ORRC.

Appendix E. Comparison with the Naive Solution

The evaluation of the runtimes and the size of the elements are given in Table 5 and Table 6 respectively. As the execution time and the size increase very quickly, we have only evaluated them for small values of parameters (ℓ_1 and ℓ_2 are bounded by 5).

TABLE 5. RUNNING TIME IN MILLISECONDS OF THE ALGORITHMS FOR THE NAIVE SCHEME. THE RESULTS ARE AN AVERAGE OVER 10 RUNS (ONLY ONE RUN FOR $(\ell_1, \ell_2) = (5, 5)$).

Scheme	Naive			
(ℓ_1, ℓ_2)	$(2, 2)$	$(3, 3)$	$(4, 4)$	$(5, 5)$
Setup	0.01	0.02	0.02	0.02
Commit	21.73	575.79	38246.20	4005869.26
VerCommit	6.10	45.92	542.97	9.42
Open	0.07	0.07	0.07	0.09
VerOpen	0.10	0.10	0.11	0.10
OpenLDP	0.23	0.34	0.42	1.51
VerOpenLDP	0.10	0.10	0.11	0.15
Gen	0.03	0.03	0.03	0.04
Sign	0.11	0.31	1.16	4.86
Ver	0.14	0.33	1.15	4.95

TABLE 6. SIZE IN BYTES OF SETUP, COMMITMENT AND PROOFS FOR THE NAIVE SCHEME

Scheme	Naive			
(ℓ_1, ℓ_2)	$(2, 2)$	$(3, 3)$	$(4, 4)$	$(5, 5)$
set	128	128	128	128
c	544	2080	8224	32800
π_*	7392	53600	410208	3212384
π	64	64	64	64
$\hat{\pi}$	64	64	64	64

According to Table 1 and Table 2, by setting $\ell_1 = \ell_2$ and setting $n = 2^{\ell_1}$, the time and size complexity of the proof π_* is in $O(n^3)$ for the naive solution and in $O(\log(n))$ for ORRC. The generation time and the size of the proof will therefore increase drastically faster with the naive solution. According to Table 5 and Table 6, it can also be seen that even for small parameters, the naive solution is already much less efficient than ORRC. For example, for $n = 2^4 = 16$, verification of π_* takes 200 times longer for the naive solution, and it is even worse concerning the generation of the proof.

Figures 18, 19 and 20 clearly illustrate the evolution of the efficiency gap between the naive solution and ORRC as the parameters grow, whether for commitment generation, commitment size or proof size.

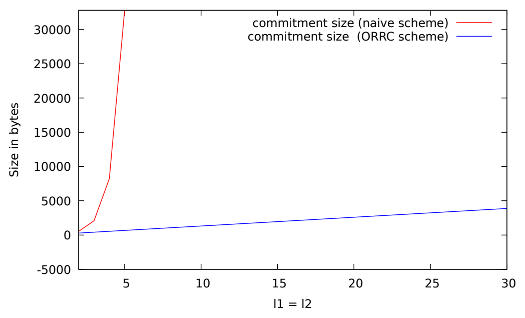

Figure 19. Comparison of the commitment size in the naive solution and ORRC.

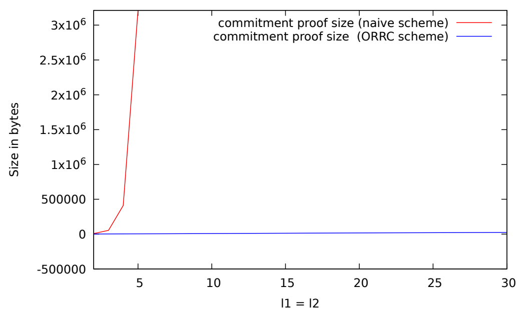

Figure 20. Comparison of the computation time for generating the commitment proof in the naive solution and ORRC.

Appendix F. Open Science Expectations

The code source of our Rust implementation is available at [37], the repository provides an implementation of our Naive and Optimized Randomized Response Commitment schemes which consists of a tuple of seven algorithms $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{VerCommit}, \mathsf{Open}, \mathsf{VerOpen}, \mathsf{OpenLDP}, \mathsf{VerOpenLDP})$. We have also implemented a Schnorr's signature that allows you to sign the generated commitment which consists of the algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Ver})$. All data used is simulated data, *i.e.* each bit of each message is sampled uniformly. To compile the code use the "cargo build –release" command in the terminal and to run the code, use the "cargo run –release" command. The run command enables the execution of the code which is located in the "main.rs" file, the program determines the computation time of each algorithm over a number of iterations. The number of iterations can be configured in the main file using the 'iter' variable. In addition, the size of the parameters ℓ_1 and ℓ_2 can be chosen by the user. The "lib.rs" file contains a serie of tests that permit the verification of the correct functionality of our algorithms, theses tests can be run with the "cargo test" command in the terminal. Our implementation is shared under the BSD 3-Clause License (the description is available at [37]). We note that portions of our code were originally derived from the curve25519-dalek implementation by Isis Agora Lovecruft and Henry de Valence which is derived from Adam Langley's Go ed25519 implementation.

Appendix G. Useful Lemmas

We first give a technical lemma, which will be useful to simplify the winning probabilities for the adversary.

Lemma 1. *Let* $\mathsf{Exp}_{\mathcal{A},b}(\lambda)$ *be an experiment depending on a bit* b *and a security parameter* λ *for an adversary* $\mathcal{A} \in \textsc{poly}(\lambda)$ *that returns a bit. we have:*

$$\left|\Pr\left[b \xleftarrow{\$} \{0,1\} \;:\; 1 \leftarrow \mathsf{Exp}_{\mathcal{A},b}(\lambda)\right] - \frac{1}{2}\right|$$
$$= \frac{1}{2}\left|\left(\Pr\left[1 \leftarrow \mathsf{Exp}_{\mathcal{A},1}(\lambda)\right] - \Pr\left[0 \leftarrow \mathsf{Exp}_{\mathcal{A},0}(\lambda)\right]\right)\right|.$$

Proof.

$$\left|\Pr\left[b \xleftarrow{\$} \{0,1\} \;:\; 1 \leftarrow \mathsf{Exp}_{\mathcal{A},b}(\lambda)\right] - \frac{1}{2}\right|$$
$$= \Bigg|\Pr\left[b \xleftarrow{\$} \{0,1\} \;:\; b = 0\right] \cdot \Pr\left[1 \leftarrow \mathsf{Exp}_{\mathcal{A},0}(\lambda)\right] +$$
$$\Pr\left[b \xleftarrow{\$} \{0,1\} \;:\; b = 1\right] \cdot \Pr\left[1 \leftarrow \mathsf{Exp}_{\mathcal{A},1}(\lambda)\right] - \frac{1}{2}\Bigg|$$
$$= \Bigg|\frac{1}{2}\left(1 - \Pr\left[0 \leftarrow \mathsf{Exp}_{\mathcal{A},0}(\lambda)\right] +\right.$$
$$\left.\Pr\left[1 \leftarrow \mathsf{Exp}_{\mathcal{A},1}(\lambda)\right]\right) - \frac{1}{2}\Bigg|$$
$$= \frac{1}{2}\left|\left(\Pr\left[1 \leftarrow \mathsf{Exp}_{\mathcal{A},1}(\lambda)\right] - \Pr\left[0 \leftarrow \mathsf{Exp}_{\mathcal{A},0}(\lambda)\right]\right)\right|.$$

□

We now give some results needed in the security analysis of our efficient LDP commitment scheme (Section 4.4).

Lemma 2. *Let* $\mathbb{G} = \langle g \rangle$ *be multiplicative group of prime order* p*. Let* $\mathcal{A}$ *be a* PPT *algorithm, consider the* LOR-DDH *experiment* $\mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{A},b}(\lambda)$ *as follows:*

```
$\mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{A},b}(\lambda)$:
$(x, y_0, y_1) \xleftarrow{\$} (\mathbb{Z}_p^*)^3$
$z \leftarrow x \cdot y_b$
$b' \leftarrow \mathcal{A}(g^x, g^{y_0}, g^{y_1}, g^z)$
return $b = b'$
```

Under the DDH assumption, there exists a negligible function $\epsilon_{\mathsf{LOR\text{-}DDH}}$ *s.t. for any* PPT *algorithm* $\mathcal{A}$*:*

$$\left|\Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{A},0}(\lambda)\right] - \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{A},1}(\lambda)\right]\right|$$
$$\leq \epsilon_{\mathsf{LOR\text{-}DDH}}(\lambda).$$

Proof. **Game** G_0: This game is the same as the LOR-DDH experiment in the case where $b \xleftarrow{\$} \{0,1\}$.

$$\Pr[\mathcal{A} \text{ wins the game } G_0]$$
$$= \Pr\left[b \xleftarrow{\$} \{0,1\} \;:\; 1 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{A},b}(\lambda)\right].$$

Game G_1: This game is the same as the game G_0 except that the challenger replaces g^z by a random element. Let $\epsilon_{\mathsf{DDH}}(\lambda)$ be the maximum DDH advantage of all PPT algorithm $\mathcal{D}$, we claim that:

$$|\Pr[\mathcal{A} \text{ wins the game } G_0] - \Pr[\mathcal{A} \text{ wins the game } G_1]|$$
$$\leq 2 \cdot \epsilon_{\mathsf{DDH}}(\lambda).$$

We prove this claim by reduction.

Assume there exists a PPT algorithm $\mathcal{A}$ such that $\epsilon_{\textsf{LOR-DDH}}$ is non negligible, we build a PPT distinguishing algorithm $\mathcal{D}$ against the DDH assumption as follows:

Algorithm $\mathcal{D}(g^x, g^y, g^\gamma)$**:** Parses g^x as X, g^y as Y_0 and g^γ as Γ, picks $r \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $Y_1 \leftarrow g^r$. $\mathcal{D}$ picks $b_* \xleftarrow{\$} \{0,1\}$, and runs $b' \leftarrow \mathcal{A}(X, Y_{b_*}, Y_{1-b_*}, \Gamma)$.

- If $b_* = b'$, then returns 1.
- Else, then returns 0.

We have:

$$\Pr[\mathcal{A} \text{ wins the game } G_0] = \Pr[(x,y) \xleftarrow{\$} (\mathbb{Z}_p^*)^2 \ : \ 1 \leftarrow \mathcal{D}(g^x, g^y, g^{x\cdot y})],$$

and

$$\Pr[\mathcal{A} \text{ wins the game } G_1] = \Pr[(x,y,z) \xleftarrow{\$} (\mathbb{Z}_p^*)^3 \ : \ 1 \leftarrow \mathcal{D}(g^x, g^y, g^z)].$$

Thus, we deduce that:

$$|\Pr[\mathcal{A} \text{ wins the game } G_0] - \Pr[\mathcal{A} \text{ wins the game } G_1]| \leq \epsilon_{\textsf{DDH}}(\lambda).$$

We claim that:

$$\Pr[\mathcal{A} \text{ wins the game } G_1] = \frac{1}{2}.$$

We note that in the game G_0, the algorithm $\mathcal{A}$ receives the tuple $(g^x, g^{y_0}, g^{y_1}, g^{x\cdot y_b})$ and it has to guess the bit b randomly picked by the challenger. The proof of the claim derives from the fact that in the game G_1, the algorithm $\mathcal{A}$ receives the tuple $(g^x, g^{y_0}, g^{y_1}, g^z)$ where g^z is a random group element, so $\mathcal{A}$ can not deduce any information on the bit b and it returns a random bit $b' \in \{0,1\}$, we have:

$$\Pr[\mathcal{A} \text{ wins the game } G_1] = \frac{1}{2}.$$

We have:

$$\begin{aligned}&\left|\Pr\left[b \xleftarrow{\$} \{0,1\} \ : \ 1 \leftarrow \textsf{Exp}^{\textsf{LOR-DDH}}_{\mathcal{A},b}(\lambda)\right] - \frac{1}{2}\right| \\ &= |\Pr[\mathcal{A} \text{ wins the game } G_0] - \\ &\quad \Pr[\mathcal{A} \text{ wins the game } G_1]| \\ &\leq \epsilon_{\textsf{DDH}}(\lambda).\end{aligned}$$

From Lemma 1, we obtain:

$$\begin{aligned}&\left|\Pr\left[b \xleftarrow{\$} \{0,1\} \ : \ 1 \leftarrow \textsf{Exp}^{\textsf{LOR-DDH}}_{\mathcal{A},b}(\lambda)\right] - \frac{1}{2}\right| \\ &= \frac{1}{2}\left|\Pr\left[1 \leftarrow \textsf{Exp}^{\textsf{LOR-DDH}}_{\mathcal{A},1}(\lambda)\right] - \right. \\ &\quad \left. \Pr\left[0 \leftarrow \textsf{Exp}^{\textsf{LOR-DDH}}_{\mathcal{A},0}(\lambda)\right]\right|.\end{aligned}$$

Finally, we have:

$$\begin{aligned}&\left|\Pr\left[1 \leftarrow \textsf{Exp}^{\textsf{LOR-DDH}}_{\mathcal{A},1}(\lambda)\right] - \right. \\ &\left. \Pr\left[0 \leftarrow \textsf{Exp}^{\textsf{LOR-DDH}}_{\mathcal{A},0}(\lambda)\right]\right| \\ &= 2 \cdot \left|\Pr\left[b \xleftarrow{\$} \{0,1\} \ : \ 1 \leftarrow \textsf{Exp}^{\textsf{LOR-DDH}}_{\mathcal{A},b}(\lambda)\right] - \frac{1}{2}\right| \\ &\leq 2 \cdot \epsilon_{\textsf{DDH}}(\lambda),\end{aligned}$$

which concludes the proof of Lemma 3. □

Lemma 3. *Let $\mathbb{G} = \langle g \rangle$ be multiplicative group of prime order p. Let $\mathcal{A}$ be a* PPT *algorithm, consider the* $\textsf{LOR-DDH}-2$ *experiment* $\textsf{Exp}^{\textsf{LOR-DDH}-2}_{\mathcal{A},b}(\lambda)$ *as follows:*

```
$\textsf{Exp}^{\textsf{LOR-DDH}-2}_{\mathcal{A},b}(\lambda)$:
$(x, y_0, y_1) \xleftarrow{\$} (\mathbb{Z}_p^*)^3$
$z_0 \leftarrow x \cdot y_b$
$z_1 \leftarrow x \cdot y_{1-b}$
$b' \leftarrow \mathcal{A}(g^x, g^{y_0}, g^{y_1}, g^{z_0}, g^{z_1})$
return $b = b'$
```

Under the DDH assumption, there exists a negligible function $\epsilon_{\textsf{LOR-DDH}-2}$ s.t. for any PPT *algorithm $\mathcal{A}$:*

$$\begin{aligned}&\left|\Pr\left[0 \leftarrow \textsf{Exp}^{\textsf{LOR-DDH}-2}_{\mathcal{A},0}(\lambda)\right] - \right. \\ &\left. \Pr\left[1 \leftarrow \textsf{Exp}^{\textsf{LOR-DDH}-2}_{\mathcal{A},1}(\lambda)\right]\right| \leq \epsilon_{\textsf{LOR-DDH}-2}(\lambda).\end{aligned}$$

Proof. **Game G_0**: This game is the same as the $\textsf{LOR-DDH}-2$ experiment in the case where $b \xleftarrow{\$} \{0,1\}$.

$$\begin{aligned}&\Pr[\mathcal{A} \text{ wins the game } G_0] \\ &= \Pr\left[b \xleftarrow{\$} \{0,1\} \ : \ 1 \leftarrow \textsf{Exp}^{\textsf{LOR-DDH}-2}_{\mathcal{A},b}(\lambda)\right].\end{aligned}$$

Game G_1: This game is the same as the game G_0 except that the challenger replaces g^{z_0} by a random element. We claim that:

$$|\Pr[\mathcal{A} \text{ wins the game } G_0] - \Pr[\mathcal{A} \text{ wins the game } G_1]| \leq \epsilon_{\textsf{DDH}}(\lambda).$$

We prove this claim by reduction.

Assume there exists a PPT algorithm $\mathcal{A}$ such that $\epsilon_{\textsf{LOR-DDH}-2}$ is non negligible, we build a PPT distinguishing algorithm $\mathcal{D}$ against the DDH assumption as follows:

Algorithm $\mathcal{D}(g^x, g^y, g^\gamma)$**:** picks $r \xleftarrow{\$} \mathbb{Z}_p^*$, Parses g^x as X, g^y as Y_0 and g^γ as Z_0, and sets $Y_1 \leftarrow g^r$ and $Z_1 \leftarrow X^r$. $\mathcal{D}$ picks $b_* \xleftarrow{\$} \{0,1\}$ and runs $b' \leftarrow \mathcal{A}(X, Y_{b_*}, Y_{1-b_*}, Z_0, Z_1)$.

- If $b_* = b'$, returns 1.
- Else, returns 0.

We have:

$$\Pr[\mathcal{A} \text{ wins the game } G_0] = \Pr[(x,y) \xleftarrow{\$} (\mathbb{Z}_p^*)^2 \ : \ 1 \leftarrow \mathcal{D}(g^x, g^y, g^{x\cdot y})],$$

and

$$\Pr[\mathcal{A} \text{ wins the game } G_1] = \Pr[(x,y,z) \xleftarrow{\$} (\mathbb{Z}_p^*)^3 \ : \ 1 \leftarrow \mathcal{D}(g^x, g^y, g^z)].$$

Thus, we deduce that:

$$|\Pr[\mathcal{A} \text{ wins the game } G_0] - \Pr[\mathcal{A} \text{ wins the game } G_1]| \leq \epsilon_{\textsf{DDH}}(\lambda),$$

Game G_2: This game is the same as the game G_1 except that the challenger replaces g^{z_1} by a random element.

Let $\epsilon_{\mathsf{DDH}}(\lambda)$ be the maximum DDH advantage of all PPT algorithm $\mathcal{D}$, we claim that:

$$|\Pr[\mathcal{A} \text{ wins the game } G_1] - \Pr[\mathcal{A} \text{ wins the game } G_2]| \leq \epsilon_{\mathsf{DDH}}(\lambda).$$

We prove this claim by reduction.

Assume there exists a PPT algorithm $\mathcal{A}$ such that $\epsilon_{\mathsf{LOR\text{-}DDH-2}}$ is non negligible, we build a PPT distinguishing algorithm $\mathcal{D}$ against the DDH assumption as follows:

Algorithm $\mathcal{D}(g^x, g^y, g^\gamma)$**:** picks $r, w \xleftarrow{\$} \mathbb{Z}_p^{*2}$, parses g^x as X, g^y as Y_0 and g^γ as Z_1 and sets $Y_1 \leftarrow g^r$ and $Z_0 \leftarrow X^w$. $\mathcal{D}$ picks $b_* \xleftarrow{\$} \{0,1\}$ and runs $b' \leftarrow \mathcal{A}(X, Y_{b_*}, Y_{1-b_*}, Z_0, Z_1)$.

- If $b_* = b'$, returns 1.
- Else, returns 0.

We have:

$$\Pr[\mathcal{A} \text{ wins the game } G_1] = \Pr[(x,y) \xleftarrow{\$} (\mathbb{Z}_p^*)^2 \;:\; 1 \leftarrow \mathcal{D}(g^x, g^y, g^{x\cdot y})],$$

and

$$\Pr[\mathcal{A} \text{ wins the game } G_2] = \Pr[(x,y,z) \xleftarrow{\$} (\mathbb{Z}_p^*)^3 \;:\; 1 \leftarrow \mathcal{D}(g^x, g^y, g^z)].$$

Thus, we deduce that:

$$|\Pr[\mathcal{A} \text{ wins the game } G_1] - \Pr[\mathcal{A} \text{ wins the game } G_2]| \leq \epsilon_{\mathsf{DDH}}(\lambda),$$

We claim that:

$$\Pr[\mathcal{A} \text{ wins the game } G_2] = \frac{1}{2}.$$

We note that in the game G_0, the algorithm $\mathcal{A}$ receives the tuple $(g^x, g^{y_0}, g^{y_1}, g^{x\cdot y_b}, g^{x\cdot y_{1-b}})$ and it has to guess the bit b randomly picked by the challenger. The proof of the claim derives from the fact that in the game G_2, the algorithm $\mathcal{A}$ receives the tuple $(g^x, g^{y_0}, g^{y_1}, g^{z_0}, g^{z_1})$ where g^{z_0} and g^{z_1} are random group elements, so $\mathcal{A}$ can not deduce any information on the bit b and it returns a random bit $b' \in \{0,1\}$, we have:

$$\Pr[\mathcal{A} \text{ wins the game } G_2] = \frac{1}{2}.$$

We obtain that:

$$\left|\Pr\left[b \xleftarrow{\$} \{0,1\} \;:\; 1 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{A},b}(\lambda)\right] - \frac{1}{2}\right| = |\Pr[\mathcal{A} \text{ wins the game } G_0] - \Pr[\mathcal{A} \text{ wins the game } G_2]| \leq 2\cdot\epsilon_{\mathsf{DDH}}(\lambda).$$

Finally, we have:

$$\left|\Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH-2}}_{\mathcal{A},1}(\lambda)\right] - \Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH-2}}_{\mathcal{A},0}(\lambda)\right]\right| = 2\cdot\left|\Pr\left[b \xleftarrow{\$} \{0,1\} \;:\; 1 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH-2}}_{\mathcal{A},b}(\lambda)\right] - \frac{1}{2}\right| \leq 4\cdot\epsilon_{\mathsf{DDH}}(\lambda),$$

which concludes the proof of the lemma. □

We recall the Discrete Logarithm (DL) assumptions in a group $\mathbb{G} = \langle g \rangle$ of prime order p. The DL assumption states that given a random $x \in \mathbb{Z}_p^*$, the probability that any PPT algorithm guesses x on input g^x is negligible.

Definition 15 (Discrete Logarithm (DL) assumption)**.** *Let $\mathbb{G} = \langle g \rangle$ be a multiplicative group of prime order p. The* Discrete Logarithm *(DL) assumption states that there exists a negligible function ϵ_{DL} such that for any* PPT *algorithm $\mathcal{A}$, we have* $\Pr\left[h \xleftarrow{\$} \mathbb{G}; x \leftarrow \mathcal{A}(h) : g^x = h\right] \leq \epsilon_{\mathsf{DL}}(\lambda)$.

Lemma 4. *Let λ be a security parameter, $\mathbb{G}$ be a group of prime order p, and $g \in \mathbb{G}$ be a generator. If the DL assumption holds, there exists a negligible function $\epsilon_{\ell\text{-}col}$ such that for all PPT algorithm $\mathcal{A} \in \textsc{poly}(\lambda)$ and all integer ℓ:*

$$\Pr\left[\begin{array}{ll} \left(g_{(i,j)}\right)_{\substack{i\in[\![\ell]\!]\\ j\in\{0,1\}}} \xleftarrow{\$} \left(\mathbb{G}^\ell\right)^2; & \left(\prod_{i=1}^{\ell} g_{(i,x_1[i])}\right. \\ (x_1,x_2) \leftarrow \mathcal{A}\left(\left(g_{(i,j)}\right)_{\substack{i\in[\![\ell]\!]\\ j\in\{0,1\}}}\right); & : \left. = \prod_{i=1}^{\ell} g_{(i,x_2[i])}\right) \\ & \wedge (x_1 \neq x_2) \end{array}\right] \leq \epsilon_{\ell\text{-}col}(\lambda).$$

Proof. We define the games G_0 and G_1 as follows:

Game G_0:
$\left(g_{(i,j)}\right)_{\substack{i\in[\![\ell]\!]\\ j\in\{0,1\}}} \xleftarrow{\$} \left(\mathbb{G}^\ell\right)^2$;
$(x_1,x_2) \leftarrow \mathcal{A}\left(\left(g_{(i,j)}\right)_{\substack{i\in[\![\ell]\!]\\ j\in\{0,1\}}}\right)$;
If $\left(\prod_{i=1}^{\ell} g_{(i,x_1[i])} = \prod_{i=1}^{\ell} g_{(i,x_2[i])}\right) \wedge (x_1 \neq x_2)$, then return 1, else return 0.

Game G_1:
$k \xleftarrow{\$} [\![\ell]\!]$; $\left(g_{(i,j)}\right)_{\substack{i\in[\![\ell]\!]\\ j\in\{0,1\}}} \xleftarrow{\$} \mathbb{G}^{2\ell}$;
$(x_1,x_2) \leftarrow \mathcal{A}\left(\left(g_{(i,j)}\right)_{\substack{i\in[\![\ell]\!]\\ j\in\{0,1\}}}\right)$;
If $x_1[k] = x_2[k]$ then abort the game and return 0;
If $\left(\prod_{i=1}^{\ell} g_{(i,x_1[i])} = \prod_{i=1}^{\ell} g_{(i,x_2[i])}\right)$, then return 1, else return 0.

We claim that, for all PPT algorithm $\mathcal{A} \in \textsc{poly}(k)$:

1) $\Pr[\mathcal{A} \text{ wins } G_0] \leq \ell \cdot \Pr[\mathcal{A} \text{ wins } G_1]$
2) $\Pr[\mathcal{A} \text{ wins } G_1] \leq \epsilon_{\mathsf{DL}}(\lambda)$

Proof of Claim 1: G_1 is defined as G_0 except that the challenger picks k at random in $[\![\ell]\!]$ and aborts if $x_1[k] = x_2[k]$. First remark that if there is no index k' such that $x_1[k'] \neq x_2[k']$, then $\mathcal{A}$ does not win the game G_0 because $x_1 = x_2$. In other words, if $\mathcal{A}$ wants to have a chance of winning, they choses x_1 and x_2 in such a way that there is at least an index k' such that $x_1[k'] \neq x_2[k']$. The game G_1 does not abort if the challenger guesses this index by picking k. Finally, The adversary increases its winning advantage by a factor equalling the probability of guessing correctly k, *i.e.* , at least $1/\ell$.

Proof of Claim 2: We prove this claim by reduction. We build an algorithm $\mathcal{B} \in \textsc{poly}(\lambda)$ that tries to return the discrete logarithm of a random group element by using $\mathcal{A}$ as a black box.

Algorithm $\mathcal{B}(h)$**:** Pick $\left(\alpha_{(i,j)}\right)_{\substack{i\in[\![\ell]\!]\\ j\in\{0,1\}}} \xleftarrow{\$} (\mathbb{Z}_p^*)^{2\ell}$ and $k \xleftarrow{\$} [\![\ell]\!]$, then set $g_{(k,0)} \leftarrow h$. For all (i,j) in $([\![\ell]\!]\times\{0,1\})\setminus\{(h,0)\}$, set $g_{i,j} \leftarrow g^{\alpha_{(i,j)}}$. Run $(x_1,x_2) \leftarrow \mathcal{A}\left(\left(g_{(i,j)}\right)_{\substack{i\in[\![\ell]\!]\\ j\in\{0,1\}}}\right)$. If $x_1[k] = x_2[k]$, then abort and return 0. If $x_1[k]=0$ then set $(x,\bar{x}) \leftarrow (x_1,x_2)$, else set $(x,\bar{x}) \leftarrow (x_2,x_1)$. We remark that $x[k]=0$ and $\bar{x}[k]=1$, so $\alpha_{(k,\bar{x}[k])}$ is defined but $\alpha_{(k,x[k])}$ is not defined. Set then return:

$$x_* \leftarrow \left(\sum_{i=1}^{\ell}\alpha_{(i,\bar{x}[i])} - \sum_{\substack{i=1\\ i\neq k}}^{\ell}\alpha_{(i,x[i])}\right)$$

We first remark that if h is chosen at random in the uniform distribution on $\mathbb{G}$, then the game G_1 is perfectly simulated for $\mathcal{A}$. If $\mathcal{A}$ wins the simulated game G_1, it holds that $\left(\prod_{i=1}^{\ell} g_{(i,x_1[i])} = \prod_{i=1}^{\ell} g_{(i,x_2[i])}\right)$ and $x_1[k] \neq x_2[k]$, which implies that $\left(\prod_{i=1}^{\ell} g_{(i,\bar{x}[i])} = g_{(k,x[k])}\cdot\prod_{\substack{i=1\\ i\neq k}}^{\ell} g_{(i,x[i])}\right)$. Moreover, we have $x[k]=0$, so $g_{(k,x[k])} = g_{(k,0)} = h$. We deduce that:

$$\begin{aligned} g^{x_*} &= g^{\left(\sum\limits_{i=1}^{\ell}\alpha_{(i,\bar{x}[i])} - \sum\limits_{\substack{i=1\\ i\neq k}}^{\ell}\alpha_{(i,x[i])}\right)} \\ &= \frac{g^{\sum\limits_{i=1}^{\ell}\alpha_{(i,\bar{x}[i])}}}{g^{\sum\limits_{\substack{i=1\\ i\neq k}}^{\ell}\alpha_{(i,x[i])}}} = \frac{\prod\limits_{i=1}^{\ell} g_{(i,\bar{x}[i])}}{\prod\limits_{\substack{i=1\\ i\neq k}}^{\ell} g_{(i,x[i])}} = h \end{aligned}$$

<u>Conclusion:</u> From the two claims, we deduce $\Pr[\mathcal{A} \text{ wins } G_0] \leq \ell\cdot\epsilon_{\mathsf{DL}}(\lambda)$, which conclude the proof since $\ell\cdot\epsilon_{\mathsf{DL}}(\lambda)$ is negligible. $\square$

G.1. Proof of Theorem 1

We show that:

$$\Pr\left[\begin{array}{l} b \xleftarrow{\$} \{0,1\}; \hat{m} \leftarrow \mathsf{LDP}(m_b); \\ b_* \leftarrow \mathsf{OptiGuess}(\lambda,\mathsf{LDP},\hat{m},m_0,m_1)\end{array} : b = b_*\right] = \frac{1}{2}\sum_{\hat{m}\in\mathcal{M}}\max(\Pr[\hat{m}\leftarrow\mathsf{LDP}(m_0)],\Pr[\hat{m}\leftarrow\mathsf{LDP}(m_1)]).$$

By abuse of notation, we will simply note $\mathsf{Pr}[b=b_*]$ for the probability:

$$\Pr\left[\begin{array}{l} b \xleftarrow{\$} \{0,1\}; \hat{m} \leftarrow \mathsf{LDP}(m_b); \\ b_* \leftarrow \mathsf{OptiGuess}(\lambda,\mathsf{LDP},\hat{m},m_0,m_1)\end{array} : b = b_*\right].$$

When b is set to a value $x\in\{0,1\}$ (resp. $\hat{m}$ to a value $\hat{x}\in\mathcal{M}$), we will note $\mathsf{Pr}[b=b_*|b=x]$ (resp. $\mathsf{Pr}[b=b_*|\hat{m}=\hat{x}]$). Since $\Pr[b=0]=\Pr[b=1]=\frac{1}{2}$, we have:

$$\begin{aligned} \Pr[b=b_*] &= \Pr[b=0]\cdot\Pr[b=b_*|b=0] \\ &\quad + \Pr[b=1]\cdot\Pr[b=b_*|b=1] \\ &= \frac{1}{2}\left(\Pr[b=b_*|b=0] + \Pr[b=b_*|b=1]\right). \end{aligned}$$

We first evaluate the first probability:

$$\begin{aligned} &\Pr[b=b_*|b=0] \\ &= \sum_{\hat{x}\in\mathcal{M}}\Pr[\hat{m}=\hat{x}|b=0]\cdot\Pr[b=b_*|b=0\wedge\hat{m}=\hat{x}] \\ &= \sum_{\hat{x}\in\mathcal{M}}\Pr[\hat{x}\leftarrow\mathsf{LDP}(m_0)]\cdot\Pr[b=b_*|b=0\wedge\hat{m}=\hat{x}] \end{aligned}$$

For all comparison operators $\square\in\{=,<,>\}$, we define the set $\mathcal{M}_\square$ as follows:

$$\hat{x}\in\mathcal{M}_\square \Leftrightarrow \Pr[\hat{x}\leftarrow\mathsf{LDP}(m_0)]\,\square\,\Pr[\hat{x}\leftarrow\mathsf{LDP}(m_1)].$$

Let $\mathcal{P}=\{\mathcal{M}_=,\mathcal{M}_<,\mathcal{M}_>\}$ be a set, we have that $\mathcal{P}$ is a partition of $\mathcal{M}$. We recall that by definition, $\mathsf{OptiGuess}$ returns 0 with probability:

- 1 if $\Pr[\hat{x}\leftarrow\mathsf{LDP}(m_0)] > \Pr[\hat{x}\leftarrow\mathsf{LDP}(m_1)]$,
- $\frac{1}{2}$ if $\Pr[\hat{x}\leftarrow\mathsf{LDP}(m_0)] = \Pr[\hat{x}\leftarrow\mathsf{LDP}(m_1)]$, and
- 0 if if $\Pr[\hat{x}\leftarrow\mathsf{LDP}(m_0)] < \Pr[\hat{x}\leftarrow\mathsf{LDP}(m_1)]$.

We deduce that:

$$\begin{aligned} &\Pr[b=b_*|b=0] \\ &= \sum_{\hat{x}\in\mathcal{M}}\Pr[\hat{x}\leftarrow\mathsf{LDP}(m_0)]\cdot\Pr[b=b_*|b=0\wedge\hat{m}=\hat{x}] \\ &= \sum_{S\in\mathcal{P}}\sum_{\hat{x}\in S}\Pr[\hat{x}\leftarrow\mathsf{LDP}(m_0)]\cdot\Pr\left[b=b_*|\begin{array}{c} b=0\wedge \\ \hat{m}=\hat{x}\end{array}\right] \\ &= \sum_{\hat{x}\in\mathcal{M}_>}\Pr[\hat{x}\leftarrow\mathsf{LDP}(m_0)] + \\ &\quad \frac{1}{2}\sum_{\hat{x}\in\mathcal{M}_=}\Pr[\hat{x}\leftarrow\mathsf{LDP}(m_0)] + \\ &0\cdot\sum_{\hat{x}\in\mathcal{M}_<}\Pr[\hat{x}\leftarrow\mathsf{LDP}(m_0)] \\ &= \sum_{\hat{x}\in\mathcal{M}_>}\Pr[\hat{x}\leftarrow\mathsf{LDP}(m_0)] + \\ &\quad \frac{1}{2}\sum_{\hat{x}\in\mathcal{M}_=}\Pr[\hat{x}\leftarrow\mathsf{LDP}(m_0)] \end{aligned}$$

In a similar way, we have:

$$\begin{aligned} &\Pr[b=b_*|b=1] \\ &= \sum_{\hat{x}\in\mathcal{M}_<}\Pr[\hat{x}\leftarrow\mathsf{LDP}(m_1)] + \frac{1}{2}\sum_{\hat{x}\in\mathcal{M}_=}\Pr[\hat{x}\leftarrow\mathsf{LDP}(m_1)]. \end{aligned}$$

Furthermore:

- for all $\hat{x}\in\mathcal{M}_>$,
$$\begin{aligned} &\max\{\Pr[\hat{x}\leftarrow\mathsf{LDP}(m_0)],\Pr[\hat{x}\leftarrow\mathsf{LDP}(m_1)]\} \\ &= \Pr[\hat{x}\leftarrow\mathsf{LDP}(m_0)]. \end{aligned}$$
- for all $\hat{x}\in\mathcal{M}_=$,
$$\begin{aligned} &\max\{\Pr[\hat{x}\leftarrow\mathsf{LDP}(m_0)],\Pr[\hat{x}\leftarrow\mathsf{LDP}(m_1)]\} \\ &= \Pr[\hat{x}\leftarrow\mathsf{LDP}(m_0)] = \Pr[\hat{x}\leftarrow\mathsf{LDP}(m_01)]. \end{aligned}$$
- for all $\hat{x}\in\mathcal{M}_<$,
$$\begin{aligned} &\max\{\Pr[\hat{x}\leftarrow\mathsf{LDP}(m_0)],\Pr[\hat{x}\leftarrow\mathsf{LDP}(m_1)]\} \\ &= \Pr[\hat{x}\leftarrow\mathsf{LDP}(m_1)]. \end{aligned}$$

We deduce:

$$\begin{aligned}
&\Pr[b = b_* | b = 0] + \Pr[b = b_* | b = 1] \\
=& \sum_{\hat{x} \in \mathcal{M}_>} \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)] + \frac{1}{2} \sum_{\hat{x} \in \mathcal{M}_=} \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)] \\
+& \sum_{\hat{x} \in \mathcal{M}_<} \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_1)] + \frac{1}{2} \sum_{\hat{x} \in \mathcal{M}_=} \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_1)] \\
=& \sum_{\hat{x} \in \mathcal{M}} \max\{\Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)], \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_1)]\}.
\end{aligned}$$

This leads to the result of the theorem:

$$\begin{aligned}
&\Pr[b = b_*] \\
=& \frac{1}{2} \sum_{\hat{x} \in \mathcal{M}} \max\{\Pr[\hat{x} \leftarrow \mathsf{LDP}(m_0)], \Pr[\hat{x} \leftarrow \mathsf{LDP}(m_1)]\}.
\end{aligned}$$

Appendix H. Proof of correctness for ORRC

Theorem 4. *The ORRC scheme is correct.*

Proof. The first condition of the correctness ($\mathsf{Open}(\mathsf{k}, \mathsf{c})$ returns m and $\mathsf{VerCommit}(\mathsf{c}, \pi_*) = \mathsf{VerOpen}(\mathsf{c}, m, \pi) = \mathsf{VerOpenLDP}(\mathsf{c}, \hat{m}, \hat{\pi}, \hat{\theta}) = 1$) follows from the construction of the protocol and has already been shown throughout Section 4.3. For the second condition, we already show that if $s = \hat{s}$ then the algorithm OpenLDP returns the original message, else it returns $t \oplus \hat{t}$. Thus, for any $\lambda \in \mathbb{N}$, any set generated from $\mathsf{Setup}(\lambda, \mathsf{LDP})$ and containing the sets $\mathcal{M} = \{0,1\}^{\ell_2}$ and $\Theta \leftarrow \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2}$, any $(m, \hat{m}_0) \in \mathcal{M}^2$, and any $\theta_0 = (s_0, t_0)$ and $\hat{\theta}_0 = (\hat{s}_0, \hat{t}_0)$ such that $(\theta_0, \hat{\theta}_0) \in \Theta^2$, we have:

$$\begin{aligned}
&\Pr[\hat{m}_1 \leftarrow \mathsf{LDP}(m) : \hat{m}_0 = \hat{m}_1] \\
&= \Pr\left[\begin{array}{l} x \overset{\$}{\leftarrow} \mathcal{M}; \\ \text{if } x = 0, \hat{m}_1 = m; \quad : \hat{m}_0 = \hat{m}_1 \\ \text{else}, \hat{m}_1 \overset{\$}{\leftarrow} \mathcal{M}; \end{array}\right]
\end{aligned}$$

$$\begin{aligned}
&\Pr\left[\begin{array}{l} \theta_1 \overset{\$}{\leftarrow} \Theta; \\ (\mathsf{k}, \mathsf{c}, \pi_*) \leftarrow \mathsf{Commit}(m, \theta_1); \quad : \hat{m}_0 = \hat{m}_1 \\ (\hat{m}_1, \hat{\pi}) \leftarrow \mathsf{OpenLDP}(\mathsf{k}, \mathsf{c}, \hat{\theta}_0) \end{array}\right] \\
&= \Pr\left[\begin{array}{l} (s_1, t_1) \overset{\$}{\leftarrow} \Theta; \\ \text{if } s_1 = \hat{s}_0, \hat{m}_1 = m; \quad : \hat{m}_0 = \hat{m}_1 \\ \text{else}, \hat{m}_1 = t_1 \oplus \hat{t}_0; \end{array}\right]
\end{aligned}$$

$$\begin{aligned}
&\Pr\left[\begin{array}{l} \hat{\theta}_1 \overset{\$}{\leftarrow} \Theta; \\ (\mathsf{k}, \mathsf{c}, \pi_*) \leftarrow \mathsf{Commit}(m, \theta_0); \quad : \hat{m}_0 = \hat{m}_1 \\ (\hat{m}_1, \hat{\pi}) \leftarrow \mathsf{OpenLDP}(\mathsf{k}, \mathsf{c}, \hat{\theta}_1) \end{array}\right] \\
&= \Pr\left[\begin{array}{l} (\hat{s}_1, \hat{t}_1) \overset{\$}{\leftarrow} \Theta; \\ \text{if } s_0 = \hat{s}_1, \hat{m}_1 = m; \quad : \hat{m}_0 = \hat{m}_1 \\ \text{else}, \hat{m}_1 = t_0 \oplus \hat{t}_1; \end{array}\right]
\end{aligned}$$

In all three cases, the generation of $\hat{m}_1$ follows the same distribution, which concludes the proof. □

Appendix I. Proof of Theorem 2

We show that the if a LDP-C scheme P is Hiding and ROS-LDP-Hiding, then P is IND-LDP-Hiding.

Proof. We define the games G_0, G_1, G_2 and G_3 as follows:

Game G_0:
```
$\mathsf{set} \leftarrow \mathsf{Setup}(\lambda, \mathsf{LDP})$
$(m_0, m_1) \leftarrow \mathcal{A}_1(\mathsf{set})$
$b' \overset{\$}{\leftarrow} \{0,1\}$
$\theta \overset{\$}{\leftarrow} \Theta$
$(\mathsf{k}, \mathsf{c}, \pi_*) \leftarrow \mathsf{Commit}(m_{b'}, \theta)$
$\hat{\theta} \leftarrow \mathcal{A}_2(\mathsf{c}, \pi_*)$
$(\hat{m}, \hat{\pi}) \leftarrow \mathsf{OpenLDP}(\mathsf{k}, \mathsf{c}, \hat{\theta})$
$b_* \leftarrow \mathcal{A}_3(\hat{m}, \hat{\pi})$
return $b_* = b'$
```

Game G_1:
```
$\mathsf{set} \leftarrow \mathsf{Setup}(\lambda, \mathsf{LDP})$
$(m_0, m_1) \leftarrow \mathcal{A}_1(\mathsf{set})$
$b' \overset{\$}{\leftarrow} \{0,1\}$
$\theta \overset{\$}{\leftarrow} \Theta$
$(\mathsf{k}, \mathsf{c}, \pi_*) \leftarrow \mathsf{Commit}(m_{b'}, \theta)$
$\hat{\theta} \leftarrow \mathcal{A}_2(\mathsf{c}, \pi_*)$
$(\hat{m}_0, \hat{\pi}_0) \leftarrow \mathsf{OpenLDP}(\mathsf{k}, \mathsf{c}, \hat{\theta})$
$\hat{\pi}_1 \leftarrow \mathsf{Sim}(\hat{m}_0)$
$b_* \leftarrow \mathcal{A}_3(\hat{m}_0, \hat{\pi}_1)$ return $b_* = b'$
```

Game G_2:
```
$\mathsf{set} \leftarrow \mathsf{Setup}(\lambda, \mathsf{LDP})$
$(m_0, m_1) \leftarrow \mathcal{A}_1(\mathsf{set})$
$b' \overset{\$}{\leftarrow} \{0,1\}$
$\theta \overset{\$}{\leftarrow} \Theta$
$(\mathsf{k}, \mathsf{c}, \pi_*) \leftarrow \mathsf{Commit}(m_{b'}, \theta)$
$\hat{\theta} \leftarrow \mathcal{A}_2(\mathsf{c}, \pi_*)$
$\hat{m}_1 \leftarrow \mathsf{LDP}(m_{b'})$
$\hat{\pi}_1 \leftarrow \mathsf{Sim}(\hat{m}_1)$
$b_* \leftarrow \mathcal{A}_3(\hat{m}_1, \hat{\pi}_1)$
return $b_* = b'$
```

Game G_3:
```
$\mathsf{set} \leftarrow \mathsf{Setup}(\lambda, \mathsf{LDP})$
$(m_0, m_1) \leftarrow \mathcal{A}_1(\mathsf{set})$
$b' \overset{\$}{\leftarrow} \{0,1\}$
$\theta \overset{\$}{\leftarrow} \Theta$
$m \overset{\$}{\leftarrow} \mathcal{M}$
$(\mathsf{k}, \mathsf{c}, \pi_*) \leftarrow \mathsf{Commit}(m, \theta)$
$\hat{\theta} \leftarrow \mathcal{A}_2(\mathsf{c}, \pi_*)$
$\hat{m}_1 \leftarrow \mathsf{LDP}(m_{b'})$
$\hat{\pi}_1 \leftarrow \mathsf{Sim}(\hat{m}_1)$
$b_* \leftarrow \mathcal{A}_3(\hat{m}_1, \hat{\pi}_1)$
return $b_* = b'$
```

We claim that, for all PPT algorithm $\mathcal{A} \in \text{POLY}(k)$:

1) $\Pr[\mathcal{A} \text{ wins the game } G_0]$
$= \Pr[\mathcal{A} \text{ wins the game } G_1]$.

2) $|\Pr[\mathcal{A} \text{ wins the game } G_1] - \Pr[\mathcal{A} \text{ wins the game } G_2]|$
$\leq \epsilon_{\text{ROS-LDP-Hiding}}(\lambda)$.

3) $|\Pr[\mathcal{A} \text{ wins the game } G_2] - \Pr[\mathcal{A} \text{ wins the game } G_3]| \leq \frac{1}{2} \cdot \epsilon_{\mathsf{Hiding}}(\lambda)$.

Proof of Claim I: Since the proof $\hat{\pi}$ is zero-knowledge, we have:

$$\Pr[\mathcal{A} \text{ wins the game } G_0] = \Pr[\mathcal{A} \text{ wins the game } G_1].$$

Proof of Claim I: We prove this claim by reduction.

```
$\underline{\mathcal{B}_1(\mathsf{set})}$:
$(m_0, m_1) \leftarrow \mathcal{A}_1(\mathsf{set})$
$b' \leftarrow \{0,1\}$
return $m_{b'}$

$\underline{\mathcal{B}_2(\mathsf{c}, \pi_*)}$:
$\hat{\theta} \leftarrow \mathcal{A}_2(\mathsf{c}, \pi_*)$
return $\hat{\theta}$

Let $b$ be the bit that $\mathcal{B}$ has to guess.
$\underline{\mathcal{B}_3(\hat{m}_b, \hat{\pi}_b)}$:
$b_* \leftarrow \mathcal{A}_3(\hat{m}_b, \hat{\pi}_b)$.
if $b_* = b'$, return $b_{**} = 1$
else, return $b_{**} = 0$
```

We have:

$$\begin{aligned}
&\Pr[\mathcal{A} \text{ wins the game } G_1] \\
&= \Pr[b_{**} = 1 | b = 0] \\
&= \Pr\left[\mathcal{B} \text{ returns } 1 \text{ in } \mathsf{Exp}^{\mathsf{ROS\text{-}LDP\text{-}Hiding}}_{P,\mathsf{LDP},\mathcal{B},0}(\lambda)\right] \\
&= \Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{ROS\text{-}LDP\text{-}Hiding}}_{P,\mathsf{LDP},\mathcal{B},0}(\lambda)\right],
\end{aligned}$$

and

$$\begin{aligned}
&\Pr[\mathcal{A} \text{ wins the game } G_2] \\
&= \Pr[b_{**} = 1 | b = 1] \\
&= \Pr\left[\mathcal{B} \text{ returns } 1 \text{ in } \mathsf{Exp}^{\mathsf{ROS\text{-}LDP\text{-}Hiding}}_{P,\mathsf{LDP},\mathcal{B},1}(\lambda)\right] \\
&= \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{ROS\text{-}LDP\text{-}Hiding}}_{P,\mathsf{LDP},\mathcal{B},1}(\lambda)\right].
\end{aligned}$$

Thus, we have:

$$\begin{aligned}
&|\Pr[\mathcal{A} \text{ wins the game } G_1] - \Pr[\mathcal{A} \text{ wins the game } G_2]| \\
&= \left|\Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{ROS\text{-}LDP\text{-}Hiding}}_{P,\mathsf{LDP},\mathcal{B},0}(\lambda)\right] - \right. \\
&\quad \left. \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{ROS\text{-}LDP\text{-}Hiding}}_{P,\mathsf{LDP},\mathcal{B},1}(\lambda)\right]\right| \\
&\leq \epsilon_{\mathsf{ROS\text{-}LDP\text{-}Hiding}}(\lambda).
\end{aligned}$$

Proof of Claim I: We prove this claim by reduction.

```
$\underline{\mathcal{B}_1(\mathsf{set})}$:
$(m_0, m_1) \leftarrow \mathcal{A}_1(\mathsf{set})$
$b' \xleftarrow{\$} \{0,1\}$
$\tilde{m}_0 \leftarrow m_{b'}$
$\tilde{m}_1 \xleftarrow{\$} \mathcal{M}$
return $(\tilde{m}_0, \tilde{m}_1)$

Let $b$ be the bit that $\mathcal{B}$ has to guess.
$\underline{\mathcal{B}_2(\mathsf{c}, \pi_*)}$:
$\hat{\theta} \leftarrow \mathcal{A}_2(\mathsf{c}, \pi_*)$
$\hat{m} \leftarrow \mathsf{LDP}(\tilde{m}_b)$
$\hat{\pi} \leftarrow \mathsf{Sim}(\hat{m})$
$b_* \leftarrow \mathcal{A}_3(\hat{m}, \hat{\pi})$
if $b_* = b'$ return $b_{**} = 1$
else, return $b_{**} = 0$
```

We have:

$$\begin{aligned}
&\Pr[\mathcal{A} \text{ wins the game } G_2] \\
&= \Pr[b_{**} = 1 | b = 0] \\
&= \Pr\left[\mathcal{B} \text{ returns } 1 \text{ in } \mathsf{Exp}^{\mathsf{Hiding}}_{P,\mathsf{LDP},\mathcal{B},0}(\lambda)\right] \\
&= \Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{Hiding}}_{P,\mathsf{LDP},\mathcal{B},0}(\lambda)\right],
\end{aligned}$$

and

$$\begin{aligned}
&\Pr[\mathcal{A} \text{ wins the game } G_3] \\
&= \Pr[b_{**} = 1 | b = 1] \\
&= \Pr\left[\mathcal{B} \text{ returns } 1 \text{ in } \mathsf{Exp}^{\mathsf{Hiding}}_{P,\mathsf{LDP},\mathcal{B},1}(\lambda)\right] \\
&= \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{Hiding}}_{P,\mathsf{LDP},\mathcal{B},1}(\lambda)\right].
\end{aligned}$$

Thus, we have:

$$\begin{aligned}
&|\Pr[\mathcal{A} \text{ wins the game } G_2] - \\
&\Pr[\mathcal{A} \text{ wins the game } G_3]| \\
&= \left|\Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{Hiding}}_{P,\mathsf{LDP},\mathcal{B},0}(\lambda)\right] - \right. \\
&\quad \left. \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{Hiding}}_{P,\mathsf{LDP},\mathcal{B},1}(\lambda)\right]\right| \\
&\leq \epsilon_{\mathsf{Hiding}}(\lambda).
\end{aligned}$$

We remark that the game G_0 is the same as the IND-LDP-Hiding experiment in the case where $b = 0$, we have:

$$\begin{aligned}
&\Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{IND\text{-}LDP\text{-}Hiding}}_{P,\mathsf{LDP},\mathcal{A},0}(\lambda)\right] \\
&= \Pr[\mathcal{A} \text{ wins the game } G_0].
\end{aligned}$$

In the game G_3, the algorithm $\mathcal{A}$ can not deduce any information on the bit b' knowing the commitment c which is the commitment of a random message, so the optimal strategy for $\mathcal{A}$ to win is to use the algorithm OptiGuess, then we have:

$$\begin{aligned}
&\Pr[\mathcal{A} \text{ wins the game } G_3] \\
&\leq \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{IND\text{-}LDP\text{-}Hiding}}_{P,\mathsf{LDP},\mathcal{A},1}(\lambda)\right].
\end{aligned}$$

Finally, we get:

$$\begin{aligned}
&\left|\Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{IND\text{-}LDP\text{-}Hiding}}_{P,\mathsf{LDP},\mathcal{A},0}(\lambda)\right] - \right. \\
&\left. \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{IND\text{-}LDP\text{-}Hiding}}_{P,\mathsf{LDP},\mathcal{A},1}(\lambda)\right]\right| \\
&= |\Pr[\mathcal{A} \text{ wins the game } G_0] - \\
&\quad \Pr[\mathcal{A} \text{ wins the game } G_3]| \\
&\leq \epsilon_{\mathsf{ROS\text{-}LDP\text{-}Hiding}}(\lambda) + \epsilon_{\mathsf{Hiding}}(\lambda).
\end{aligned}$$

□

Appendix J. Proof of Theorem 3 (Security Proofs for ORRC)

The proof of Theorem 3 for the scheme ORRC follows from the following lemmas.

Lemma 5. *The ORRC scheme instantiated with zeronowledge proofs and a group where the DDH assumption holds is hiding.*

Lemma 6. *The ORRC scheme instantiated with zeronowledge proofs and a group where the DDH assumption holds is ROS-LDP-hiding.*

Lemmas 5 and 6 implies that the ORRC scheme is IND-LDP-hiding.

Lemma 7. *The ORRC scheme instantiated with extractable proofs and a group where the DDH assumption holds is binding.*

Lemma 8. *The ORRC scheme instantiated with extractable proofs and a group where the DDH assumption holds is LDP-binding.*

Lemma 9. *The ORRC scheme instantiated with extractable proofs and a group where the DDH assumption holds is probabilistic-LDP-binding.*

J.1. Proof of Lemma 5 for ORRC

An adversary $\mathcal{A}$ wins the Hiding experiment by distinguishing between two messages which one has been committed. In other words, since our scheme, we use an encoding based on the discrete logarithm assumption for the commitment, that means the adversary wins the experiment by deducing the original message from the encoding. To prove the lemma, we use the following sequence of games [28] based on indistinguishability, where the first game is the Hiding experiment given in the case where $b = 0$ and the last game is the Hiding experiment in the case where $b = 1$.

Proof. **Game G_0**: This game is the same as the Hiding experiment in the case where $b = 0$, we have:

$$\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_0] = \Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{Hiding}}_{\mathsf{ORRC},\mathsf{LDP},\mathcal{A},0}(\lambda)\right].$$

Game G_1: This game is the same as the game G_0 except that the challenger replaces the zero-knowledge proof π_* by a simulated proof, we have:

$$\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_0] = \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_1].$$

In the game G_1, the challenger commits the message m_0, we parse the input of the adversary c as $(y, (A_{(i,1)})_{i\in[\![\ell_1]\!]}, (A_{(i,2)}, B_{(i,0)}, B_{(i,1)})_{i\in[\![\ell_2]\!]}$. We note that the information of the i-th bit of the message m_0 is contained in $A_{(i,2)}$.
Game G_2: This game is the same as the Hiding experiment in the case where $b = 1$, we have:

$$\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_2] = \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{Hiding}}_{\mathsf{ORRC},\mathsf{LDP},\mathcal{A},1}(\lambda)\right].$$

To prove indistinguishability between G_1 and G_2, we consider a sequence of games where we use the hybrid argument [33] and in which we replace for each $i \in [\![\ell_2]\!]$, $A_{(i,2)} = f^x_{i,m_0[i]}$ by $f^x_{i,m_1[i]}$.
Game $G_{1,k}$ (for all $k \in [\![\ell_2]\!]$): We define $G_{1,0}$ as G_1 and G_{1,ℓ_2} as G_2. This game is the same as the game $G_{1,k-1}$ except that the challenger replaces $A_{(k,2)} = f^x_{k,m_0[k]}$ by $f^x_{k,m_1[k]}$ in the commitment, we claim that:

$$|\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1}] - \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k}]| \leq \tfrac{1}{2}\epsilon_{\mathsf{LOR\text{-}DDH}}(\lambda).$$

We prove this claim by reduction, we build the following algorithm $\mathcal{B}$ against the LOR-DDH experiment:
Algorithm $\mathcal{B}(g^x, g^{y_0}, g^{y_1}, g^z)$: generates a multiplicative group $\mathbb{G} = \langle g \rangle$ of prime order p, $\mathcal{M} \leftarrow \{0,1\}^{\ell_2}$ and $\Theta \leftarrow \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2}$, picks $(u_{i,j})_{\substack{i\in[\![\ell_1]\!]\\ j\in\{0,1\}}} \xleftarrow{\$} (\mathbb{Z}_p^*)^{\ell_1}$ and $(v_{i,j}, w_{i,j})_{\substack{i\in[\![\ell_2]\!]\\ j\in\{0,1\}}} \xleftarrow{\$} ((\mathbb{Z}_p^*)^{\ell_2})^2$. It sets $\mathsf{set} = (\mathbb{G}, p, (g^{u_{i,j}})_{\substack{i\in[\![\ell_1]\!]\\ j\in\{0,1\}}}, (g^{v_{i,j}}, g^{w_{i,j}})_{\substack{i\in[\![\ell_2]\!]\\ j\in\{0,1\}}}, \mathcal{M}, \Theta)$ except that it replaces $g^{v_{k,0}}$ by g^{y_0} and $g^{v_{k,1}}$ by g^{y_1}. $\mathcal{B}$ generates $\theta \xleftarrow{\$} \Theta$, parses θ as (s,t), and runs $(m_0, m_1) \leftarrow \mathcal{A}_1(\mathsf{set})$. $\mathcal{B}$ sets $\mathsf{c} = (g^x, ((g^x)^{u_{i,s[i]}}))_{i\in[\![\ell_1]\!]}, ((g^x)^{v_{i,m_0[i]}}, (g^x)^{w_{i,t[i]}}, (g^x)^{w_{i,1-t[i]}})_{i\in[\![\ell_2]\!]})$ except that for any $i < k$, replaces $(g^x)^{v_{i,m_0[i]}}$ by $(g^x)^{v_{i,m_1[i]}}$, and replaces $(g^x)^{v_{k,m_0[k]}}$ by g^z. $\mathcal{B}$ generates a simulated zero-knowledge proof π_* and runs $b' \leftarrow \mathcal{A}(\mathsf{set}, \mathsf{c}, \pi_*)$.

- If $m_0[k] = 0$ and $m_1[k] = 1$:
  - If $b' = 0$, then $g^z = g^{x\cdot y_{m_0[k]}} = g^{x\cdot y_0}$.
  - If $b' = 1$, then $g^z = g^{x\cdot y_{m_1[k]}} = g^{x\cdot y_1}$.

  $\mathcal{B}$ returns b'.
- Else, $(m_0[k] = 1$ and $m_1[k] = 0)$:
  - If $b' = 0$, then $g^z = g^{x\cdot y_{m_0[k]}} = g^{x\cdot y_1}$.
  - If $b' = 1$, then $g^z = g^{x\cdot y_{m_1[k]}} = g^{x\cdot y_0}$.

  $\mathcal{B}$ returns $1 - b'$.

We have:

1) $\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1} | m_0[k] = m_1[k]]$
$= \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k} | m_0[k] = m_1[k]]$.

2) $\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1} | m_0[k] = 0 \wedge m_1[k] = 1]$
$= \Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{B},0}(\lambda)\right]$,
and
$\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k} | m_0[k] = 0 \wedge m_1[k] = 1]$
$= \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{B},1}(\lambda)\right]$.

This leads to:

$$|\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1} | m_0[k] = 0 \wedge m_1[k] = 1] - \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k} | m_0[k] = 0 \wedge m_1[k] = 1]|$$
$$= \left|\Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{B},0}(\lambda)\right] - \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{B},1}(\lambda)\right]\right|$$
$$\leq \epsilon_{\mathsf{LOR\text{-}DDH}}(\lambda).$$

3) $\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1} | m_0[k] = 1 \wedge m_1[k] = 0]$
$= \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{B},0}(\lambda)\right]$
$= 1 - \Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{B},0}(\lambda)\right]$,
and

$$\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k} | m_0[k] = 1 \wedge m_1[k] = 0]$$
$$= \Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{B},1}(\lambda)\right]$$
$$= 1 - \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{B},1}(\lambda)\right].$$

This leads to:

$$|\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1} | m_0[k] = 1 \wedge m_1[k] = 0] - \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k} | m_0[k] = 1 \wedge m_1[k] = 0]|$$
$$= \left|\Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{B},1}(\lambda)\right] - \Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{B},0}(\lambda)\right]\right|$$
$$\leq \epsilon_{\mathsf{LOR\text{-}DDH}}(\lambda).$$

We obtain:

$$|\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1}] - \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k}]|$$
$$= |\Pr[m_0[k] = m_1[k]] \cdot \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1} | m_0[k] = m_1[k]] + \Pr[m_0[k] \neq m_1[k]] \cdot \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1} | m_0[k] \neq m_1[k]] - (\Pr[m_0[k] = m_1[k]] \cdot \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k} | m_0[k] = m_1[k]] + \Pr[m_0[k] \neq m_1[k]] \cdot \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k} | m_0[k] \neq m_1[k]])|$$
$$= \tfrac{1}{2} \cdot |\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1} | m_0[k] \neq m_1[k]] - \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k} | m_0[k] \neq m_1[k]]|$$
$$\leq \tfrac{1}{2} \cdot \epsilon_{\mathsf{LOR\text{-}DDH}}(\lambda).$$

Thus, we get:

$$|\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_1] - \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_2]| \leq \tfrac{\ell_2}{2} \cdot \epsilon_{\mathsf{LOR\text{-}DDH}}(\lambda).$$

Finally, we have:

$$\left|\Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{Hiding}}_{\mathsf{ORRC},\mathsf{LDP},\mathcal{A},0}(\lambda)\right] - \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{Hiding}}_{\mathsf{ORRC},\mathsf{LDP},\mathcal{A},1}(\lambda)\right]\right|$$
$$= |\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_0] - \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_2]|$$
$$\leq \tfrac{\ell_2}{2} \cdot \epsilon_{\mathsf{LOR\text{-}DDH}}(\lambda),$$

which concludes the proof of the lemma since $\frac{\ell_2}{2} \cdot \epsilon_{\mathsf{LOR\text{-}DDH}}(\lambda)$ is negligible. □

J.2. Proof of Lemma 6 for ORRC

An adversary $\mathcal{A}$ wins the ROS-LDP-Hiding experiment by distinguishing between a message ouputted by opening the commitment and a message from the distribution $\mathcal{D}$ (the same distribution as the distribution given by LDP mechanism in Section 3). In this case, we set $n_1 = 2^{\ell_1}$ and $n_2 = 2^{\ell_2}$, the LDP mechanism defines that the probability of having the same message is $\frac{2^{\ell_1}+2^{\ell_2}-1}{2^{\ell_1+\ell_2}}$ and the probability of having other messages is $\frac{2^{\ell_1}-1}{2^{\ell_1+\ell_2}}$. We define the algorithm LDP which output a value picks in the distribution $\mathcal{D}$ as follows:

$\mathsf{LDP}(m)$: picks $s, t \xleftarrow{\$} \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2}$, if $s = 0$, sets $\hat{m} = m$, else, sets $\hat{m} = t$ and returns $\hat{m}$.

$\mathsf{Sim}(\hat{m})$: simulates the proof $\hat{\pi}$ and returns $\hat{\pi}$.

To prove the lemma, we set the first game as the ROS-LDP-Hiding experiment in the case where $b = 0$ and the last game as the ROS-LDP-Hiding experiment in the case where $b = 1$.

Proof. **Game G_0**: This game is the same as the ROS-LDP-Hiding experiment in the case $b = 0$, we have:

$$\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_0] = \Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{ROS\text{-}LDP\text{-}Hiding}}_{\mathsf{ORRC},\mathsf{LDP},\mathcal{A},0}(\lambda)\right].$$

Game G_1: This game is the same as the game G_0 except that the challenger replaces the proofs $\hat{\pi}$ and π_* by simulated proofs, we have:

$$\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_1] = \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_0].$$

From the game G_1, we parse θ as (s_0, t_0), the input of the adversary c as $(y, (A_{(i,1)})_{i \in [\![\ell_1]\!]}, (A_{(i,2)}, B_{(i,0)}, B_{(i,1)})_{i \in [\![\ell_2]\!]})$ and $\hat{\theta}$ as $(\hat{s}, \hat{t})$. We note that if $s_0 = \hat{s}$, then $\hat{m} = m$, else $\hat{m} = t_0 \oplus \hat{t}$. (s_0, t_0) being unknown to the adversary, the information of the i-th bit of the message $\hat{m}_0$ in the commitment c is contained in the i-th bit of the secrets s_0 and t_0 which is hidden in $A_{(i,1)}$ and, respectively $B_{(i,0)}$ and $B_{(i,1)}$.

Game G_2: This game is the same as the game G_1 except that the challenger replaces for all $i \in [\![\ell_1]\!]$, $A_{i,1} = g^x_{i,s_0[i]}$ by $g^x_{i,s_1[i]}$.

To prove indistinguishability, we pick $(s_1, t_1) \xleftarrow{\$} \Theta$ and between G_1 and G_2, we consider a sequence of games where we use the hybrid argument [33] and in which we replace for each $i \in [\![\ell_1]\!]$, $A_{(i,1)} = g^x_{i,s_0[i]}$ by $g^x_{i,s_1[i]}$ in the commitment.

Game $G_{1,k}$(for all $k \in [\![\ell_1]\!]$): We define $G_{1,0}$ as G_1 and G_{1,ℓ_1} as G_2. This game is the same as the game $G_{1,k-1}$ except that the challenger replaces $A_{(k,1)} = g^x_{k,s_0[k]}$ by $g^x_{k,s_1[k]}$, we claim that:

$$|\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1}] - \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k}]| \leq \tfrac{1}{2}\epsilon_{\mathsf{LOR\text{-}DDH}}(\lambda).$$

We prove this claim by reduction. We build the following PPT algorithm against the LOR-DDH experiment:

Algorithm $\mathcal{B}(g^x, g^{y_0}, g^{y_1}, g^{x \cdot y_b})$: generates a multiplicative group $\mathbb{G} = \langle g \rangle$ of prime order p, $\mathcal{M} \leftarrow \{0,1\}^{\ell_2}$, $\Theta \leftarrow \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2}$ and $\theta \xleftarrow{\$} \Theta$, parses θ as (s_0, t_0), picks $(u_{i,j})_{\substack{i \in [\![\ell_1]\!] \\ j \in \{0,1\}}}$ and $(v_{i,j}, w_{i,j})_{\substack{i \in [\![\ell_2]\!] \\ j \in \{0,1\}}} \xleftarrow{\$} ((\mathbb{Z}_p^*)^2)^{\ell_2}$. It sets $\mathsf{set} = (\mathbb{G}, p, (g^{u_{i,j}})_{\substack{i \in [\![\ell_1]\!] \\ j \in \{0,1\}}}, (g^{v_{i,j}}, g^{w_{i,j}})_{\substack{i \in [\![\ell_2]\!] \\ j \in \{0,1\}}}, \mathcal{M}, \Theta)$ except that it replaces $g^{u_{k,0}}$ by g^{y_0} and $g^{u_{k,1}}$ by g^{y_1}. Runs $r \leftarrow \mathcal{A}_1(\mathsf{set})$ and generates a tuple $\mathsf{c} = (g^x, ((g^x)^{u_{i,s_0[i]}})_{i \in [\![\ell_1]\!]}, ((g^x)^{v_{i,m[i]}}, (g^x)^{w_{i,t_0[i]}}, (g^x)^{w_{i,1-t_0[i]}})_{i \in [\![\ell_2]\!]})$ (as in the game G_0). Picks $(s_1, t_1) \xleftarrow{\$} \theta$, for any $i < k$, replaces $(g^x)^{u_{i,s_0[i]}}$ by $(g^x)^{u_{i,s_1[i]}}$ and it replaces $(g^x)^{u_{k,s_0[k]}}$ by $g^{x \cdot y_b}$. $\mathcal{B}$ generates a simulated proof π_* and runs $\hat{\theta} \leftarrow \mathcal{A}_2(\mathsf{set}, m, \mathsf{c}, \pi_*)$. Parses $\hat{\theta}$ as $(\hat{s}, \hat{t})$. If $\hat{s} = s_0$, sets $\hat{m} = m$, else, sets $\hat{m} = t_0 \oplus \hat{t}$. $\mathcal{B}$ generates a simulated proof $\hat{\pi}$ and runs $b' \leftarrow \mathcal{A}_3(\mathsf{set}, m, \mathsf{c}, \pi_*, \hat{\theta}, \hat{m}, \hat{\pi})$.

- If $s_0[k] = 0$:
  - If $b' = 0$, then $g^{x \cdot y_b} = g^{x \cdot y_{s_0[k]}} = g^{x \cdot y_0}$.
  - Else, $(b' = 1)$, then $g^{x \cdot y_b} = g^{x \cdot y_{s_1[k]}} = g^{x \cdot y_1}$.

$\mathcal{B}$ returns b'.

- Else, $(s_0[k] = 1)$:
  - If $b' = 0$, then $g^{x\cdot y_b} = g^{x\cdot y_{s_0[k]}} = g^{x\cdot y_1}$.
  - Else, $(b' = 1)$, then $g^{x\cdot y_b} = g^{x\cdot y_{s_1[k]}} = g^{x\cdot y_0}$.

  $\mathcal{B}$ returns $1 - b'$.

We have:

1) $\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1} | s_0[k] = s_1[k]]$
$= \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k} | s_0[k] = s_1[k]]$.

2) $\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1} | s_0[k] = 0 \wedge s_1[k] = 1]$
$= \Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{B},0}(\lambda)\right]$,
and
$\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k} | s_0[k] = 0 \wedge s_1[k] = 1]$
$= \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{B},1}(\lambda)\right]$.

This leads to:

$$\begin{aligned}&|\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1} | s_0[k] = 0 \wedge s_1[k] = 1] - \\ &\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k} | s_0[k] = 0 \wedge s_1[k] = 1]| \\ &= \left|\Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{B},0}(\lambda)\right] - \right. \\ &\quad \left.\Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{B},1}(\lambda)\right]\right| \\ &\leq \epsilon_{\mathsf{LOR\text{-}DDH}}(\lambda).\end{aligned}$$

3) $\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1} | s_0[k] = 1 \wedge s_1[k] = 0]$
$= \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{B},0}(\lambda)\right]$
$= 1 - \Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{B},0}(\lambda)\right]$,
and
$\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k} | s_0[k] = 1 \wedge s_1[k] = 0]$
$= \Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{B},1}(\lambda)\right]$
$= 1 - \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{B},1}(\lambda)\right]$.

This leads to:

$$\begin{aligned}&|\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1} | s_0[k] = 1 \wedge s_1[k] = 0] - \\ &\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k} | s_0[k] = 1 \wedge s_1[k] = 0]| \\ &= \left|\Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{B},1}(\lambda)\right] - \right. \\ &\quad \left.\Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH}}_{\mathcal{B},0}(\lambda)\right]\right| \\ &\leq \epsilon_{\mathsf{LOR\text{-}DDH}}(\lambda).\end{aligned}$$

We obtain:

$$\begin{aligned}&|\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1}] - \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k}]| \\ &= |\Pr[s_0[k] = s_1[k]] \\ &\quad \cdot \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1} | s_0[k] = s_1[k]] + \\ &\quad \Pr[s_0[k] \neq s_1[k]] \\ &\quad \cdot \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1} | s_0[k] \neq s_1[k]] - \\ &\quad (\Pr[s_0[k] = s_1[k]] \\ &\quad \cdot \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k} | s_0[k] = s_1[k]] + \\ &\quad \Pr[s_0[k] \neq s_1[k]] \\ &\quad \cdot \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k} | s_0[k] \neq s_1[k]])| \\ &= \tfrac{1}{2}|\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k-1} | s_0[k] \neq s_1[k]] - \\ &\quad \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{1,k} | s_0[k] \neq s_1[k]]| \\ &\leq \tfrac{1}{2}\epsilon_{\mathsf{LOR\text{-}DDH}}(\lambda).\end{aligned}$$

Thus, we deduce that:

$$\begin{aligned}&|\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_1] - \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_2]| \\ &\leq \tfrac{\ell_1}{2} \cdot \epsilon_{\mathsf{LOR\text{-}DDH}}(\lambda).\end{aligned}$$

Game G_3: This game is the same as the game G_2 except that the challenger replaces for all $i \in [\![\ell_2]\!]$, $\left(B_{(i,0)}, B_{(i,1)}\right) = \left(h^x_{i,t_0[i]}, h^x_{i,1\oplus t_0[i]}\right)$ by $\left(h^x_{i,t_1[i]}, h^x_{i,1\oplus t_1[i]}\right)$, we have:

$$\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_3] = \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{ROS\text{-}LDP\text{-}Hiding}}_{\mathsf{ORRC,LDP},\mathcal{A},1}(\lambda)\right].$$

To prove indistinguishability between G_2 and G_3, we consider a sequence of games where we use the hybrid argument [33] and in which we replace for each $i \in [\![\ell_2]\!]$, $\left(B_{(i,0)}, B_{(i,1)}\right) = \left(h^x_{i,t_0[i]}, h^x_{i,1\oplus t_0[i]}\right)$ by $\left(h^x_{i,t_1[i]}, h^x_{i,1\oplus t_1[i]}\right)$.

Game $G_{2,k}$(for all $k \in [\![\ell_2]\!]$): We define $G_{2,0}$ as G_2 and G_{2,ℓ_2} as G_3. This game is the same as the game $G_{2,k-1}$ except that the challenger replaces $\left(B_{(k,0)}, B_{(k,1)}\right) = (h^x_{k,t_0[k]}, h^x_{k,1\oplus t_0[k]})$ by $(h^x_{k,t_1[k]}, h^x_{k,1\oplus t_1[k]})$, we claim that:

$$\begin{aligned}&|\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k-1}] - \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k}]| \\ &\leq \tfrac{1}{2}\epsilon_{\mathsf{LOR\text{-}DDH}-2}(\lambda).\end{aligned}$$

We prove this claim by reduction. We build the following PPT algorithm against the $\mathsf{LOR\text{-}DDH} - 2$ experiment:

Algorithm $\mathcal{B}(g^x, g^{y_0}, g^{y_1}, g^{x\cdot y_b}, g^{x\cdot y_{1-b}})$**:** generates a multiplicative group $\mathbb{G} = \langle g \rangle$ of prime order p, $\mathcal{M} \leftarrow \{0,1\}^{\ell}_2$, $\Theta \leftarrow \{0,1\}^{\ell_1} \times \{0,1\}^{\ell_2}$ and $\theta \xleftarrow{\$} \Theta$. Parses θ as (s_0, t_0), picks $(u_{i,j})_{\substack{i\in[\![\ell_1]\!]\\ j\in\{0,1\}}}$ and $(v_{i,j}, w_{i,j})_{\substack{i\in[\![\ell_2]\!]\\ j\in\{0,1\}}} \xleftarrow{\$} ((\mathbb{Z}^*_p)^2)^l$. It sets $\mathsf{set} = (\mathbb{G}, p, (g^{u_{i,j}})_{\substack{i\in[\![\ell_1]\!]\\ j\in\{0,1\}}}, (g^{v_{i,j}}, g^{w_{i,j}})_{\substack{i\in[\![\ell_2]\!]\\ j\in\{0,1\}}}, \mathcal{M}, \Theta)$ except that it replaces $g^{w_{k,0}}$ by g^{y_0} and $g^{w_{k,1}}$ by g^{y_1}. Runs $m \leftarrow \mathcal{A}_1(\mathsf{set})$, picks $(s_1, t_1) \xleftarrow{\$} \theta$, and generates a tuple $\mathsf{c} = (g^x, ((g^x)^{u_{i,s_1[i]}})_{i\in[\![\ell_1]\!]}, ((g^x)^{v_{i,m[i]}}, (g^x)^{w_{i,t_0[i]}}, (g^x)^{w_{i,1-t_0[i]}})_{i\in[\![\ell_2]\!]})$ (as in the game G_1). For any $i < k$, replaces $(g^x)^{w_{i,t_0[i]}}$ by $(g^x)^{w_{i,t_1[i]}}$, $(g^x)^{w_{i,1\oplus t_0[i]}}$ by $(g^x)^{w_{i,1\oplus t_1[i]}}$ and it replaces $(g^x)^{w_{k,t_0[k]}}$ by $g^{x\cdot y_b}$ and $(g^x)^{w_{k,1\oplus t_0[k]}}$ by $g^{x\cdot y_{1-b}}$. $\mathcal{B}$ generates a simulated proof π_* and runs $\hat{\theta} \leftarrow \mathcal{A}_2(\mathsf{set}, m, \mathsf{c}, \pi_*)$. Parses $\hat{\theta}$ as $(\hat{s}, \hat{t})$.

- If $\hat{s} = s_0$, sets $\hat{m} = m$.

- Else, sets $\hat{m} = t_0 \oplus \hat{t}$.

$\mathcal{B}$ generates a simulated proof $\hat{\pi}$ and runs $b' \leftarrow \mathcal{A}_3(\mathsf{set}, m, \mathsf{c}, \pi_*, \hat{\theta}, \hat{m}, \hat{\pi})$.

- If $t_0[k] = 0$:
  - If $b' = 0$, then $g^{x \cdot y_b} = g^{x \cdot y_{t_0[k]}} = g^{x \cdot y_0}$.
  - If $b' = 1$, then $g^{x \cdot y_b} = g^{x \cdot y_{1 \oplus \hat{t}_0[k]}} = g^{x \cdot y_1}$.

  $\mathcal{B}$ returns b'.
- Else ($t_0[k] = 1$):
  - If $b' = 0$, then $g^{x \cdot y_b} = g^{x \cdot y_{t_0[k]}} = g^{x \cdot y_1}$.
  - If $b' = 1$, then $= g^{x \cdot y_{1 \oplus t_0[k]}} = g^{x \cdot y_0}$.

  $\mathcal{B}$ returns $1 - b'$.

We have:

1) $\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k-1} | t_0[k] = t_1[k]]$
$= \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k} | t_0[k] = t_1[k]]$.

2) $\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k-1} | t_0[k] = 0 \wedge t_1[k] = 1]$
$= \Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH-2}}_{\mathcal{B},0}(\lambda)\right]$,
and
$\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k} | t_0[k] = 0 \wedge t_1[k] = 1]$
$= \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH-2}}_{\mathcal{B},1}(\lambda)\right]$.

This lead to:

$$\begin{aligned}&|\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k-1} | t_0[k] = 0 \wedge t_1[k] = 1] - \\ &\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k} | t_0[k] = 0 \wedge t_1[k] = 1]| \\ &= \left|\Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH-2}}_{\mathcal{B},0}(\lambda)\right] - \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH-2}}_{\mathcal{B},1}(\lambda)\right]\right| \\ &\leq \epsilon_{\mathsf{LOR\text{-}DDH-2}}(\lambda).\end{aligned}$$

3) $\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k-1} | t_0[k] = 1 \wedge t_1[k] = 0]$
$= \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH-2}}_{\mathcal{B},0}(\lambda)\right]$
$= 1 - \Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH-2}}_{\mathcal{B},0}(\lambda)\right]$,
and
$\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k} | t_0[k] = 1 \wedge t_1[k] = 0]$
$= \Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH-2}}_{\mathcal{B},1}(\lambda)\right]$
$= 1 - \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH-2}}_{\mathcal{B},1}(\lambda)\right]$.

This leads to:

$$\begin{aligned}&|\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k-1} | t_0[k] = 1 \wedge t_1[k] = 0] - \\ &\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k} | t_0[k] = 1 \wedge t_1[k] = 0]| \\ &= \left|\Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH-2}}_{\mathcal{B},1}(\lambda)\right] - \Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{LOR\text{-}DDH-2}}_{\mathcal{B},0}(\lambda)\right]\right| \\ &\leq \epsilon_{\mathsf{LOR\text{-}DDH-2}}(\lambda).\end{aligned}$$

We obtain:

$$\begin{aligned}&|\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k-1}] - \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k}]| \\ &= |\Pr[t_0[k] = t_1[k]] \\ &\quad \cdot \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k-1} | t_0[k] = t_1[k]] + \\ &\quad \Pr[t_0[k] \neq t_1[k]] \\ &\quad \cdot \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k-1} | t_0[k] \neq t_1[k]] - \\ &\quad (\Pr[t_0[k] = t_1[k]] \\ &\quad \cdot \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k-1} | t_0[k] = t_1[k]] + \\ &\quad \Pr[t_0[k] \neq t_1[k]] \\ &\quad \cdot \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k-1} | t_0[k] \neq t_1[k])| \\ &= \tfrac{1}{2} \cdot |\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k-1} | t_0[k] \neq t_1[k]] - \\ &\quad \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_{2,k} | t_0[k] \neq t_1[k]]| \\ &\leq \tfrac{1}{2} \cdot \epsilon_{\mathsf{LOR\text{-}DDH-2}}(\lambda).\end{aligned}$$

Thus, we deduce that:

$$\begin{aligned}&|\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_2] - \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_3]| \\ &\leq \tfrac{\ell_2}{2} \cdot \epsilon_{\mathsf{LOR\text{-}DDH-2}}(\lambda).\end{aligned}$$

Finally, we have:

$$\begin{aligned}&\left|\Pr\left[0 \leftarrow \mathsf{Exp}^{\mathsf{ROS\text{-}LDP\text{-}Hiding}}_{\mathsf{ORRC,LDP},\mathcal{A},0}(\lambda)\right] - \right. \\ &\left.\Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{ROS\text{-}LDP\text{-}Hiding}}_{\mathsf{ORRC,LDP},\mathcal{A},1}(\lambda)\right]\right| \\ &= |\Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_0] - \Pr[\mathcal{A} \text{ returns } 1 \text{ in } G_3]| \\ &\leq \tfrac{\ell_1}{2} \cdot \epsilon_{\mathsf{LOR\text{-}DDH}}(\lambda) + \tfrac{\ell_2}{2} \cdot \epsilon_{\mathsf{LOR\text{-}DDH-2}}(\lambda),\end{aligned}$$

which concludes the proof of the lemma since $\frac{\ell_1}{2} \cdot \epsilon_{\mathsf{LOR\text{-}DDH}}(\lambda)$ and $\frac{\ell_2}{2} \cdot \epsilon_{\mathsf{LOR\text{-}DDH-2}}(\lambda)$ are negligible. □

J.3. Proof of Lemma 7 for ORRC

An adversary $\mathcal{A}$ wins the Binding experiment when it can open the commitment in two different ways by using the Open algorithm. To prove the lemma, we use the following sequence of games [28] based on failure events, where the first game is the Binding experiment and the last game is the experiment in which the adversary has no chance of winning.

Proof. **Game G_0**: This game is the same as the Binding experiment, we have:

$$\Pr[\mathcal{A} \text{ wins the game } G_0] = \Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{Binding}}_{\mathsf{ORRC,LDP},\mathcal{A}}(\lambda)\right].$$

Game G_1: This game is the same as the game G_0 except that the challenger uses the extractors on the zero-knowledge proofs π_0 and π_1 outputted by the adversary and aborts and returns 0 on the event F_1 = "An extractor fails on at least one proof".

We note $\mathcal{C}$ the challenger in the game G_1. $\mathcal{C}$ extracts the witness x from the proofs π_0 and π_1. We emphasize that if $\mathcal{A}$ wins its game, then the two witnesses are correctly extracted and these witnesses are the same because the proofs implicitly use the same $y = g^x$. $\mathcal{C}$ parses c as $(y, (A_{(i,1)})_{i \in [\![\ell_1]\!]}, (A_{(i,2)}, B_{(i,0)}, B_{(i,1)})_{i \in [\![\ell_2]\!]})$ and computes $A_2 = \prod_{i=1}^{\ell_2} A_{(i,2)}$, $\alpha_{2,0} = \prod_{i=1}^{\ell_2} f_{i,m_0[i]}$ and $\alpha_{2,1} = \prod_{i=1}^{\ell_2} f_{i,m_1[i]}$.

If the proofs are correctly extracted, we have:
$y = g^x \wedge A_2 = \alpha_{2,0}^x$ and $y = g^x \wedge A_2 = \alpha_{2,1}^x$.

Let $\epsilon_{\mathsf{ext}}(\lambda)$ be the maximum of failure probability of the extractors, we have:

$$|\Pr[\mathcal{A} \text{ wins the game } G_0] - \Pr[\mathcal{A} \text{ wins the game } G_1]| \leq \Pr[F_1] \leq 2 \cdot \epsilon_{\mathsf{ext}}(\lambda).$$

From the game G_1, an adversary can win the experiment by returning $(m_0, m_1, \pi_0, \pi_1, \mathsf{c})$ such that $m_0 \neq m_1$ and $\prod_{i=1}^{\ell_2} f_{i,m_0[i]} = \prod_{i=1}^{\ell_2} f_{i,m_1[i]}$.

Game G_2: This game is the same as the game G_1 except that aborts and returns 0 on the event F_2 = "$\mathcal{A}$ outputs $(m_0, m_1, \pi_0, \pi_1, \mathsf{c})$ such that $\prod_{i=1}^{\ell_2} f_{i,m_0[i]} = \prod_{i=1}^{\ell_2} f_{i,m_1[i]}$ and $m_0 \neq m_1$", we have:

$$|\Pr[\mathcal{A} \text{ wins the game } G_1] = \Pr[\mathcal{A} \text{ wins the game } G_2]| \leq \Pr[F_2].$$

If the event F_2 occurs, then the adversary $\mathcal{A}$ build $(m_0, m_1, \pi_0, \pi_1, \mathsf{c})$ such that π_0 and π_1 are valid proofs and the adversary wins the experiment.

We claim that:

$$\Pr[F_2] \leq \epsilon_{\ell_2\text{-col}}(\lambda).$$

We prove this claim by reduction. Assume the event F_2 occurs with non negligible probability. We build the following PPT algorithm $\mathcal{B}$ that play the experiment defined in the lemma 4:

Algorithm $\mathcal{B}\left((\widetilde{g}_{(i,j)})_{\substack{i\in[\![\ell_2]\!]\\ j\in\{0,1\}}}\right)$**:** simulates the game G_1 to $\mathcal{A}$ except that during the setup generation, for each $i \in [\![\ell_2]\!]$, it replaces $f_{i,0}$ by $\widetilde{g}_{(i,0)}$, and $f_{i,1}$ by $\widetilde{g}_{(i,1)}$. Runs $(m_0, m_1, \pi_0, \pi_1, \mathsf{c}) \leftarrow \mathcal{A}(\mathsf{set})$. $\mathcal{B}$ computes $\prod_{i=1}^{\ell_2} f_{i,m_0[i]}$ and $\prod_{i=1}^{\ell_2} f_{i,m_1[i]}$, if the event F_2 does not happen *i.e.* if $\prod_{i=1}^{\ell_2} f_{i,m_0[i]} \neq \prod_{i=1}^{\ell_2} f_{i,m_1[i]}$ or $m_0 = m_1$, aborts the experiment, else it returns (m_0, m_1).

If $\mathcal{A}$ wins the game G_1 and F_2 occurs, then $\mathcal{B}$ returns $(x_1, x_2) \in \left(\{0,1\}^{\ell_2}\right)^2$ s.t. $x_1 \neq x_2$ and $\prod_{i=1}^{\ell_2} \widetilde{g}_{(i,x_1[i])} = \prod_{i=1}^{\ell_2} \widetilde{g}_{(i,x_2[i])}$, which happens with negligible probability $\epsilon_{\ell_2\text{-col}}$ according to the lemma 4.

Thus, we deduce that:

$$|\Pr[\mathcal{A} \text{ wins the game } G_1] - \Pr[\mathcal{A} \text{ wins the game } G_2]| \leq \Pr[F_2] \leq \epsilon_{\ell_2\text{-col}}(\lambda).$$

If F_1 and F_2 does not happen, then the values $A_{(i,2)}$ are correct and there is no product collision for two different messages, so in the game G_2, the adversary has no possibility to win, finally, we have:

$$\Pr\left[1 \leftarrow \mathsf{Exp}_{\mathsf{ORRC},\mathsf{LDP},\mathcal{A}}^{\mathsf{Binding}}(\lambda)\right] = |\Pr[\mathcal{A} \text{ wins the game } G_0] - \Pr[\mathcal{A} \text{ wins the game } G_2]| \leq 2 \cdot \epsilon_{\mathsf{ext}}(\lambda) + \epsilon_{\ell_2\text{-col}}(\lambda).$$

which concludes the proof of the lemma since $2 \cdot \epsilon_{\mathsf{ext}}(\lambda)$ and $2 \cdot \epsilon_{\ell_2\text{-col}}(\lambda)$ are negligible. □

J.4. Proof of Lemma 8 for ORRC

An adversary $\mathcal{A}$ wins the LDP-Binding experiment by opening the commitment in two different ways using the same seed on OpenLDP algorithm. To prove the lemma, we use the following sequence of games [28] based on failure events, where the first game is the LDP-Binding experiment and the last game is the experiment in which the adversary has no chance of winning.

Proof. **Game** G_0: This game is the same as the LDP-Binding experiment, we have:

$$\Pr[\mathcal{A} \text{ wins the game } G_0] = \Pr\left[1 \leftarrow \mathsf{Exp}_{\mathsf{ORRC},\mathsf{LDP},\mathcal{A}}^{\mathsf{LDP\text{-}Binding}}(\lambda)\right].$$

Game G_1: This game is the same as the game G_0 except that the challenger uses the extractors on the zero-knowledge proofs $\hat{\pi}_0$, $\hat{\pi}_1$ and π_* outputted by the adversary, and aborts and returns 0 on the event F_1 = "An extractor fails on at least one proof".

We note $\mathcal{C}$ the challenger in the game G_1. Parses π_* as $(\pi_{A_1}, \pi_{A_2}, \pi_B)$, $\mathcal{C}$ extracts the witnesses from the proofs $\hat{\pi}_0$, $\hat{\pi}_1$, π_{A_1}, π_{A_2} and π_B. We emphasize that if $\mathcal{A}$ wins its game, then the five witnesses are correctly extracted and these witnesses are the same because the proofs implicitly use the same $y = g^x$. $\mathcal{C}$ parses c as $(y, (A_{(i,1)})_{i\in[\![\ell_1]\!]}, (A_{(i,2)}, B_{(i,0)}, B_{(i,1)})_{i\in[\![\ell_2]\!]})$ and $\hat{\theta}$ as $(\hat{s}, \hat{t})$.

$\mathcal{C}$ computes $A_2 = \prod_{i=1}^{\ell_2} A_{(i,2)}$, $\alpha_{2,0} = \prod_{i=1}^{\ell_2} f_{i,\hat{m}_0[i]}$, $\alpha_{2,1} = \prod_{i=1}^{\ell_2} f_{i,\hat{m}_1[i]}$, $A_1 = \prod_{i=1}^{\ell_1} A_{(i,1)}$, $\alpha_1 = \prod_{i=1}^{\ell_1} g_{i,\hat{s}[i]}$, $\beta_0 = \prod_{i=1}^{\ell_2} B_{(i,\hat{m}_0[i])}$, $\beta_1 = \prod_{i=1}^{\ell_2} B_{(i,\hat{m}_1[i])}$ and $\gamma = \prod_{i=1}^{\ell_2} h_{i,\hat{t}[i]}$.

If the proofs are correctly extracted,

- From the proof $\hat{\pi}_0$, we have:
$$(y = g^x \wedge A_1 = \alpha_1^x \wedge A_2 = \alpha_{2,0}^x) \vee (y = g^x \wedge A_1 \neq \alpha_1^x \wedge \beta_0 = \gamma^x),$$
- From the proof $\hat{\pi}_1$, we have:
$$(y = g^x \wedge A_1 = \alpha_1^x \wedge A_2 = \alpha_{2,1}^x) \vee (y = g^x \wedge A_1 \neq \alpha_1^x \wedge \beta_1 = \gamma^x).$$
- From the proof π_{A_1}, we have:
$$\bigwedge_{i=1}^{\ell_1}\left(y = g^x \wedge \bigvee_{j=0}^{1} A_{(i,1)} = g_{i,j}^x\right),$$
- From the proof π_{A_2}, we have:
$$\bigwedge_{i=1}^{\ell_2}\left(y = g^x \wedge \bigvee_{j=0}^{1} A_{(i,2)} = f_{i,j}^x\right),$$
- And from the proof π_B, we have:
$$\bigwedge_{i=1}^{\ell_2}\left(y = g^x \wedge \left(\bigwedge_{j=0}^{1} B_{(i,j)} = h_{i,j}^x \vee \bigwedge_{j=0}^{1} B_{(i,j)} = h_{i,1-j}^x\right)\right).$$

Let $\epsilon_{\mathsf{ext}}(\lambda)$ be the maximum of failure probability of the extractors, we have:

$$|\Pr[\mathcal{A} \text{ wins the game } G_0] - \Pr[\mathcal{A} \text{ wins the game } G_1]| \leq \Pr[F_1] \leq 5 \cdot \epsilon_{\mathsf{ext}}(\lambda).$$

$\hat{s}$ and $(A_{(i,1)})_{i \in [\![\ell_1]\!]}$ being given and fixed by the adversary, either we have $A_1 = \alpha_1^x$, orthe̲rwise we have $A_1 \neq \alpha_1^x$.

We remark that in the case where we have $A_1 = \alpha_1^x$, an adversary wins the experiment by returning $\hat{m}_0$ and $\hat{m}_1$ such that π_*, $\hat{\pi}_0$ and $\hat{\pi}_1$ are valid proofs and $\prod_{i=1}^{\ell_2} f_{i,\hat{m}_0[i]} = \prod_{i=1}^{\ell_2} f_{i,\hat{m}_1[i]}$ and $\hat{m}_0 \neq \hat{m}_1$.

In the case where we have $A_1 \neq \alpha_1^x$, an adversary wins the experiment by returning:

- $\hat{m}_0$ and $\hat{m}_1$ such that π_*, $\hat{\pi}_0$, and $\hat{\pi}_1$ are valid proofs and $\prod_{i=1}^{\ell_2} B_{(i,\hat{m}_0[i])} = \prod_{i=1}^{\ell_2} B_{(i,\hat{m}_1[i])} \left(= \prod_{i=1}^{\ell_2} h^x_{i,\hat{t}[i]} \right)$ and $\hat{m}_0 \neq \hat{m}_1$.
- $\hat{m}_0$ and $\hat{m}_1$ such that π_*, $\hat{\pi}_0$, and $\hat{\pi}_1$ are valid proofs and $\prod_{i=1}^{\ell_2} h_{(i,\hat{m}_0[i] \oplus t[i])} = \prod_{i=1}^{\ell_2} h_{(i,\hat{m}_1[i] \oplus t[i])}$ and $\hat{m}_0 \neq \hat{m}_1$.

Game G_2: This game is the same as the game G_1 except that aborts and returns 0 on the event F_2 = "$\mathcal{A}$ returns $(\hat{m}_0, \hat{m}_1, \hat{\pi}_0, \hat{\pi}_1, \pi_*, \mathsf{c}, \hat{\theta})$ such that $\prod_{i=1}^{\ell_2} f_{i,\hat{m}_0[i]} = \prod_{i=1}^{\ell_2} f_{i,\hat{m}_1[i]}$ and $\hat{m}_0 \neq \hat{m}_1$", we have:

$$|\Pr[\mathcal{A} \text{ wins the game } G_1] = \Pr[\mathcal{A} \text{ wins the game } G_2]| \leq \Pr[F_2] \leq \epsilon_{\ell_2\text{-col}}(\lambda).$$

If the event F_2 occurs, then the adversary $\mathcal{A}$ returns $(\hat{m}_0, \hat{m}_1, \hat{\pi}_0, \hat{\pi}_1, \pi_*, \mathsf{c}, \hat{\theta})$ such that $\hat{\pi}_0$, $\hat{\pi}_1$ and π_* are valid proofs, then the adversary wins the experiment.

We claim that:

$$\Pr[F_2] \leq \epsilon_{\ell_2\text{-col}}(\lambda).$$

We prove this claim by reduction. Assume the event F_2 occurs with non negligible probability, we build the following PPT algorithm $\mathcal{B}$ that play the experiment defined in the lemma 4:

Algorithm $\mathcal{B}\left((\widetilde{g}_{(i,j)})_{i \in [\![\ell_2]\!], j \in \{0,1\}}\right)$**:** simulates the game G_1 to $\mathcal{A}$ except that during the setup generation, for each $i \in [\![\ell_2]\!]$, it replaces $f_{i,0}$ by $\widetilde{g}_{(i,0)}$, and $f_{i,1}$ by $\widetilde{g}_{(i,1)}$. Runs $(\hat{m}_0, \hat{m}_1, \hat{\pi}_0, \hat{\pi}_1, \pi_*, \mathsf{c}, \hat{\theta}) \leftarrow \mathcal{A}(\mathsf{set})$. $\mathcal{B}$ computes $\prod_{i=1}^{\ell_2} f_{i,\hat{m}_0[i]}$ and $\prod_{i=1}^{\ell_2} f_{i,\hat{m}_1[i]}$, if the event F_2 does not happen *i.e.* if $\prod_{i=1}^{\ell_2} f_{i,\hat{m}_0[i]} \neq \prod_{i=1}^{\ell_2} f_{i,\hat{m}_1[i]}$ or $\hat{m}_0 = \hat{m}_1$, aborts the experiment, else it returns $(\hat{m}_0, \hat{m}_1)$.

If $\mathcal{A}$ wins the game G_1 and F_2 occurs, then $\mathcal{B}$ returns $x_1, x_2 \in \{0,1\}^{\ell_2}$ s.t. $x_1 \neq x_2$ and $\prod_{i=1}^{\ell_2} \widetilde{g}_{(i,x_1[i])} = \prod_{i=1}^{\ell_2} \widetilde{g}_{(i,x_2[i])}$, which happens with non negligible probability $\epsilon_{\ell_2\text{-col}}$ according to the lemma 4.

Thus, we have:

$$|\Pr[\mathcal{A} \text{ wins the game } G_1] - \Pr[\mathcal{A} \text{ wins the game } G_2]| \leq \Pr[F_2] \leq \epsilon_{\ell_2\text{-col}}(\lambda).$$

Game G_3: This game is the same as the game G_2 except that aborts and returns 0 on the event F_3 = "$\mathcal{A}$ returns $(\hat{m}_0, \hat{m}_1, \hat{\pi}_0, \hat{\pi}_1, \pi_*, \mathsf{c}, \hat{\theta})$ such that $\prod_{i=1}^{\ell_2} h_{i,t[i] \oplus \hat{m}_0[i]} = \prod_{i=1}^{\ell_2} h_{i,t[i] \oplus \hat{m}_1[i]}$ and $\hat{m}_0 \neq \hat{m}_1$ and $s \neq \hat{s}$", we have:

$$|\Pr[\mathcal{A} \text{ wins the game } G_2] = \Pr[\mathcal{A} \text{ wins the game } G_3]| \leq \Pr[F_3] \leq \epsilon_{\ell_2\text{-col}}(\lambda).$$

If the event F_3 occurs, then the adversary $\mathcal{A}$ returns $(\hat{m}_0, \hat{m}_1)$ such that $\hat{\pi}_0$, $\hat{\pi}_1$ and π_* are valid proofs and the adversary wins the experiment.

We claim that:

$$\Pr[F_3] \leq \epsilon_{\ell_2\text{-col}}(\lambda).$$

The reduction from the game G_2 to game G_3 is similar as the reduction from the game G_1 to the game G_2:

Algorithm $\mathcal{B}\left((\widetilde{g}_{(i,j)})_{\substack{i \in [\![\ell_2]\!] \\ j \in \{0,1\}}}\right)$**:** simulates the game G_2 to $\mathcal{A}$ except that during the setup generation, for each $i \in [\![\ell_2]\!]$, it replaces $h_{(i,0)}$ by $\widetilde{g}_{(i,0)}$, and $h_{(i,1)}$ by $\widetilde{g}_{(i,1)}$.

Runs $(\hat{m}_0, \hat{m}_1, \hat{\pi}_0, \hat{\pi}_1, \pi_*, \mathsf{c}, \hat{\theta}) \leftarrow \mathcal{A}(\mathsf{set})$. $\mathcal{B}$ uses the extractor on the proof π_*, we note x the witness extracted.

For each $i \in [\![\ell_1]\!]$,

- If $A_{(i,1)} = g^x_{i,0}$, sets $s[i] = 0$,
- Else sets $s[i] = 1$.

For each $i \in [\![\ell_2]\!]$,

- If $B_{(i,0)} = h^x_{i,0}$, sets $t[i] = 0$,
- Else sets $t[i] = 1$.

$\mathcal{B}$ computes $\prod_{i=1}^{\ell_2} h_{i,t[i] \oplus \hat{m}_0[i]}$ and $\prod_{i=1}^{\ell_2} h_{i,t[i] \oplus \hat{m}_1[i]}$. If the event F_3 does not happen, aborts the experiment, else it returns $(t \oplus \hat{m}_0, t \oplus \hat{m}_1)$.

If $\mathcal{A}$ wins the game G_2 and F_3 occurs, then $\mathcal{B}$ returns $x_1, x_2 \in \{0,1\}^{\ell_2}$ s.t. $x_1 \neq x_2$ and $\prod_{i=1}^{\ell_2} \widetilde{g}_{(i,x_1[i])} = \prod_{i=1}^{\ell_2} \widetilde{g}_{(i,x_2[i])}$, which happens with non negligible probability $\epsilon_{\ell_2\text{-col}}$ according to the lemma 4.

Thus, we have:

$$|\Pr[\mathcal{A} \text{ wins the game } G_2] = \Pr[\mathcal{A} \text{ wins the game } G_3]| \leq \Pr[F_3] \leq \epsilon_{\ell_2\text{-col}}(\lambda).$$

Game G_4: This game is the same as the game G_3 except that aborts and returns 0 on the event F_4 = "$\mathcal{A}$ returns $(\hat{m}_0, \hat{m}_1)$ such that $\prod_{i=1}^{\ell_2} B_{(i,\hat{m}_0[i])} = \prod_{i=1}^{\ell_2} B_{(i,\hat{m}_1[i])}$ and $\hat{m}_0 \neq \hat{m}_1$ and $s \neq \hat{s}$".

If the event F_4 occurs, then the adversary $\mathcal{A}$ returns $(\hat{m}_0, \hat{m}_1, \hat{\pi}_0, \hat{\pi}_1, \pi_*, \mathsf{c}, \hat{\theta})$ such that $(\hat{\pi}_0, \hat{\pi}_1)$ are valid proofs, then the adversary wins the experiment.

We claim that:

$$\Pr[F_4] \leq \epsilon_{\ell_2\text{-col}}(\lambda).$$

The reduction from the game G_3 to game G_4 is similar as the reduction from the game G_1 to the game G_2:

Algorithm $\mathcal{B}\left((\widetilde{g}_{(i,j)})_{\substack{i\in[\![\ell_2]\!]\\ j\in\{0,1\}}}\right)$**:** simulates the game G_3 to $\mathcal{A}$ except that during the setup generation, for each $i \in [\![\ell_2]\!]$, it replaces $h_{i,0}$ by $\widetilde{g}_{(i,0)}$, and $h_{i,1}$ by $\widetilde{g}_{(i,1)}$. Runs $(\hat{m}_0, \hat{m}_1, \hat{\pi}_0, \hat{\pi}_1, \pi_*, \mathsf{c}, \hat{\theta}) \leftarrow \mathcal{A}(\mathsf{set})$. $\mathcal{B}$ computes $\prod_{i=1}^{\ell_2} B_{(i,\hat{m}_0[i])}$ and $\prod_{i=1}^{\ell_2} B_{(i,\hat{m}_1[i])}$. $\mathcal{B}$ uses the extractor on the proof π_*, we note x the witness extracted.

For each $i \in [\![\ell_1]\!]$,

- If $A_{(i,1)} = g^x_{i,0}$, sets $s[i] = 0$,
- Else sets $s[i] = 1$.

For each $i \in [\![\ell_2]\!]$,

- If $B_{(i,0)} = h^x_{i,0}$, sets $t[i] = 0$,
- Else sets $t[i] = 1$.

If the event F_4 does not happen, aborts the experiment, else it returns $(\hat{m}_0 \oplus t, \hat{m}_1 \oplus t)$.

If $\mathcal{A}$ wins the game G_3 and F_4 occurs, then $\mathcal{B}$ returns $x_1, x_2 \in \{0,1\}^{\ell_2}$ s.t. $x_1 \neq x_2$ and $\prod_{i=1}^{\ell_2} \widetilde{g}_{(i,x_1[i])} = \prod_{i=1}^{\ell_2} \widetilde{g}_{(i,x_2[i])}$, which happens with non negligible probability $\epsilon_{\ell_2\text{-col}}$ according to the lemma 4.

Thus, we have:

$$\begin{aligned}&|\Pr[\mathcal{A} \text{ wins the game } G_3] - \Pr[\mathcal{A} \text{ wins the game } G_4]|\\ &\leq \Pr[F_4] \leq \epsilon_{\ell_2\text{-col}}(\lambda).\end{aligned}$$

If F_1, F_2, F_3 and F_4 does not happen, then the commitment is well-formed and there is no product collision for two different messages on the α_2, β and γ, which implies the uniqueness of the open message for $(\hat{s}, \hat{t})$, so in the game G_4, the adversary has no possibility of winning, finally, we have:

$$\begin{aligned}&\Pr\left[1 \leftarrow \mathsf{Exp}^{\mathsf{LDP\text{-}Binding}}_{\mathsf{ORRC},\mathsf{LDP},\mathcal{A}}(\lambda)\right]\\ &= |\Pr[\mathcal{A} \text{ wins the game } G_0] - \Pr[\mathcal{A} \text{ wins the game } G_4]|\\ &\leq 5 \cdot \epsilon_{\mathsf{ext}}(\lambda) + 3 \cdot \epsilon_{\ell_2\text{-col}}(\lambda).\end{aligned}$$

which concludes the proof of the lemma since $5 \cdot \epsilon_{\mathsf{ext}}(\lambda)$ and $3 \cdot \epsilon_{\ell_2\text{-col}}(\lambda)$ are negligible. □

J.5. Proof of lemma 9 for ORRC

The Probabilistic-LDP-binding security is unachieved when the adversary returns an opened message which does not follow the LDP mechanism distribution according to the seed and the original message.To prove the lemma, we use the following sequence of games [28] based on failure events, where the first game is the Prob-LDP-Binding experiment and the last game is the experiment in which the adversary cannot returning a message that does not follow the same distribution as the distribution of the LDP mechanism.

Proof. **Game** G_0: This game is the same as the Prob-LDP-Binding experiment in the case where $b = 0$, we have:

$$\forall \hat{m} \in \mathcal{M},$$
$$\Pr[\hat{m} \leftarrow G_0] = \Pr\left[\hat{m} \leftarrow \mathsf{Exp}^{\mathsf{Prob\text{-}LDP\text{-}Binding}}_{\mathsf{ORRC},\mathsf{LDP},\mathcal{A},0}(\lambda)\right].$$

Game G_1: From the game G_0, parses the output of $\mathcal{A}_1$ as $(\mathsf{c}, \pi_*, m, \pi)$, and the output of $\mathcal{A}_2$ as $(\hat{m}_0, \hat{\pi})$. Parses π_* as $(\pi_{A_1}, \pi_{A_2}, \pi_B)$. This game is the same as the game G_0 except that the challenger uses the extractors on the zero-knowledge proofs $\pi_{A_1}, \pi_{A_2}, \pi_B$, π and $\hat{\pi}$, and aborts and returns 0 on the event F_1 = "An extractor fails on at least one proof", we have:

$$\forall \hat{m} \in \mathcal{M},$$
$$|\Pr[\hat{m} \leftarrow G_0] - \Pr[\hat{m} \leftarrow G_1]| \leq \Pr[F_1].$$

We note $\mathcal{C}$ the challenger in the game G_1. $\mathcal{C}$ extracts the witnesses from the proofs π_{A_1}, π_{A_2}, π_B, π and $\hat{\pi}$. We emphasize that if the proofs are verified, then the witnesses are correctly extracted and these witnesses are the same because the proofs implicitly used the same $y = g^x$.

$\mathcal{C}$ parses c as $(y, (A_{(i,1)})_{i\in[\![\ell_1]\!]}, (A_{(i,2)}, B_{(i,0)}, B_{(i,1)})_{i\in[\![\ell_2]\!]})$ and $\hat{\theta}$ as $(\hat{s}, \hat{t})$. $\mathcal{C}$ computes $A_2 = \prod_{i=1}^{\ell_2} A_{(i,2)}$, $\alpha_2 = \prod_{i=1}^{\ell_2} f_{i,m[i]}$, $\alpha_2' = \prod_{i=1}^{\ell_2} f_{i,\hat{m}_0[i]}$, $A_1 = \prod_{i=1}^{\ell_1} A_{(i,1)}$, $\alpha_1 = \prod_{i=1}^{\ell_1} g_{i,\hat{s}[i]}$, $\beta = \prod_{i=1}^{\ell_2} B_{(i,\hat{m}_0[i])}$ and $\gamma = \prod_{i=1}^{\ell_2} h_{i,\hat{t}[i]}$.

The challenger $\mathcal{C}$ finds the tuple (s,t) used by the adversary $\mathcal{A}$ in the commitment c as follows:

For each $i \in [\![\ell_1]\!]$, and $k \in [\![\ell_2]\!]$

- If $A_{(i,1)} = g^x_{i,0}$, sets $s[i] = 0$, else sets $s[i] = 1$.
- If $B_{(k,0)} = h^x_{k,0}$, sets $t[k] = 0$, else sets $t[k] = 1$.

If the proofs are correctly extracted,

- From the proof π_{A_1}, we have:

$$\bigwedge_{i=1}^{\ell_1}\left(y = g^x \wedge \bigvee_{j=0}^{\ell_1} A_{(i,1)} = g^x_{i,j}\right),$$

- From the proof π_{A_2}, we have:

$$\bigwedge_{i=1}^{\ell_2}\left(y = g^x \wedge \bigvee_{j=0}^{\ell_2} A_{(i,2)} = f^x_{i,j}\right),$$

- From the proof π_B, we have:

$$\bigwedge_{i=1}^{\ell_2}\left(y = g^x \wedge \left(\bigwedge_{j=0}^{\ell_2} B_{(i,j)} = h^x_{i,j} \vee \bigwedge_{j=0}^{\ell_2} B_{(i,j)} = h^x_{i,1-j}\right)\right),$$

- From the proof π, we have:

$$y = g^x \wedge A_2 = \alpha_2^x$$

- And from the proof $\hat{\pi}$, we have:

$$(y = g^x \wedge A_1 = \alpha_1^x \wedge A_2 = \alpha_2'^x) \\ \vee (y = g^x \wedge A_1 \neq \alpha_1^x \wedge \beta = \gamma^x)\,.$$

Let ϵ_{ext} be the maximum on the failure probability of the extractors, we have:
$\forall \hat{m} \in \mathcal{M}$,

$$|\Pr[\hat{m} \leftarrow G_0] - \Pr[\hat{m} \leftarrow G_1]| \leq \Pr[F_1] \leq 5 \cdot \epsilon_{\mathsf{ext}}(\lambda).$$

An adversary may attempt to biases the LDP mechanism by returning a commitment $\mathsf{c} = (y, (A_{(i,1)})_{i\in[\![\ell_1]\!]}, (A_{(i,2)}, B_{(i,0)}, B_{(i,1)})_{i\in[\![\ell_2]\!]})$, zero-knowledge proofs π_* and $\hat{\pi}$, and the messages m and $\hat{m}_0$ such that:

- $\prod_{i=1}^{\ell_1} A_{(i,1)} = \prod_{i=1}^{\ell_1} g_{i,s[i]}^x = \prod_{i=1}^{\ell_1} g_{i,\hat{s}[i]}^x$ and $s \neq \hat{s}$.
- $\prod_{i=1}^{\ell_2} f_{i,m[i]} = \prod_{i=1}^{\ell_2} f_{i,\hat{m}_0[i]}$ and $\hat{m}_0 \neq m$.
- $\prod_{i=1}^{\ell_2} h_{i,\hat{m}_0[i]\oplus t[i]} = \prod_{i=1}^{\ell_2} h_{i,\hat{t}[i]}$ and $\hat{m}_0 \oplus t \neq \hat{t}$.

Game G_2: This game is the same as the game G_1 except that the challenger aborts and returns 0 on the event $F_2 =$ "The algorithm $\mathcal{A}_1$ returns $(\mathsf{c}, \pi_*, m, \pi)$ and $\mathcal{A}_2$ returns $(\hat{m}_0, \hat{\pi})$ such that $\prod_{i=1}^{\ell_2} f_{i,\hat{m}_0[i]} = \prod_{i=1}^{\ell_2} f_{i,m[i]}$ and $\hat{m}_0 \neq m$", we have:
$\forall \hat{m} \in \mathcal{M}$,

$$|\Pr[\hat{m} \leftarrow G_1] - \Pr[\hat{m} \leftarrow G_2]| \leq \Pr[F_2].$$

We claim that:

$$\Pr[F_2] \leq \epsilon_{\ell_2\text{-col}}(\lambda).$$

We prove this claim by reduction.
Assume the event F_2 occurs with non negligible probability, we build the following PPT algorithm $\mathcal{B}$ that play the experiment defined in the lemma 4:
Algorithm $\mathcal{B}\left((\widetilde{g}_{(i,j)})_{i\in[\![\ell_2]\!],j\in\{0,1\}}\right)$**:** simulates the game G_2 to $\mathcal{A}$ except that during the setup generation, for each $i \in [\![\ell_2]\!]$, it replaces $f_{i,0}$ by $\widetilde{g}_{(i,0)}$, and $f_{i,1}$ by $\widetilde{g}_{(i,1)}$. Runs $(\mathsf{c}, \pi_*, m, \pi) \leftarrow \mathcal{A}_1(\mathsf{set})$, picks $\hat{\theta} \xleftarrow{\$} \Theta$ and runs $(\hat{m}_0, \hat{\pi}) \leftarrow \mathcal{A}_2(\hat{\theta})$. $\mathcal{B}$ computes $\prod_{i=1}^{\ell_2} f_{i,\hat{m}_0[i]}$ and $\prod_{i=1}^{\ell_2} f_{i,m[i]}$, if the event F_2 does not happen *i.e.* if $\prod_{i=1}^{\ell_2} f_{i,\hat{m}_0[i]} \neq \prod_{i=1}^{\ell_2} f_{i,m[i]}$ or $\hat{m}_0 = m$, aborts the experiment, else it returns $(\hat{m}_0, m)$.

If the experiment returns $\hat{m}_0$ in G_1 and F_2 happens, then $\mathcal{B}$ returns $x_1, x_2 \in \{0,1\}^{\ell_2}$ s.t. $x_1 \neq x_2$ and $\prod_{i=1}^{\ell_2} \widetilde{g}_{(i,x_1[i])} = \prod_{i=1}^{\ell_2} \widetilde{g}_{(i,x_2[i])}$, which happens with non negligible probability $\epsilon_{\ell_2\text{-col}}$ according to the lemma 4.
Thus, the following holds:
$\forall \hat{m} \in \mathcal{M}$,

$$|\Pr[\hat{m} \leftarrow G_1] - \Pr[\hat{m} \leftarrow G_2]| \leq \Pr[F_2] \leq \epsilon_{\ell_2\text{-col}}(\lambda).$$

Game G_3: Note that in the case where $\hat{s} \neq s$, we have $\hat{m} = t \oplus \hat{t}$, which implies $h_{i,\hat{t}[i]}^x = h_{i,\hat{m}[i]\oplus t[i]}^x$ and $B_{(i,\hat{m}[i])} = h_{i,\hat{t}[i]}^x$. This game is the same as the game G_3 except that aborts and returns 0 on the event $F_3 =$ "The algorithm $\mathcal{A}_1$ returns $(\mathsf{c}, \pi_*, m, \pi)$ and $\mathcal{A}_2$ returns $(\hat{m}_0, \hat{\pi})$ such that $\prod_{i=1}^{\ell_2} h_{i,\hat{m}_0[i]\oplus t[i]} = \prod_{i=1}^{\ell_2} h_{i,\hat{t}[i]}$ and $\hat{m}_0 \oplus t \neq \hat{t}$ and $s \neq \hat{s}$", we have:
$\forall \hat{m} \in \mathcal{M}$,

$$|\Pr[\hat{m} \leftarrow G_2] - \Pr[\hat{m} \leftarrow G_3]| \leq \Pr[F_3].$$

We claim that:

$$\Pr[F_3] \leq \epsilon_{\ell_2\text{-col}}(\lambda).$$

We prove this claim by reduction.
Assume the event F_3 occurs with non negligible probability, we build the following PPT algorithm $\mathcal{B}$ that play the experiment defined in the lemma 4:
Algorithm $\mathcal{B}\left((\widetilde{g}_{(i,j)})_{i\in[\![\ell_2]\!],j\in\{0,1\}}\right)$**:** simulates the game G_2 to $\mathcal{A}$ except that during the setup generation, for each $i \in [\![\ell_2]\!]$, it replaces $h_{i,0}$ by $\widetilde{g}_{(i,0)}$, and $h_{i,1}$ by $\widetilde{g}_{(i,1)}$. Runs $(\mathsf{c}, \pi_*, m, \pi) \leftarrow \mathcal{A}_1(\mathsf{set})$, picks $\hat{\theta} \xleftarrow{\$} \Theta$ and runs $(\hat{m}_0, \hat{\pi}) \leftarrow \mathcal{A}_2(\hat{\theta})$. $\mathcal{B}$ uses the extractor on the proof π_*, we note x the witness extracted.
For each $i \in [\![\ell_2]\!]$,

- If $B_{(i,0)} = h_{i,0}^x$, sets $t[i] = 0$.
- Else sets $t[i] = 1$.

$\mathcal{B}$ computes $\prod_{i=1}^{\ell_2} h_{i,\hat{m}_0[i]\oplus t[i]}$ and $\prod_{i=1}^{\ell_2} h_{i,\hat{t}[i]}$, if the event F_3 does not happen *i.e.* if $\prod_{i=1}^{\ell_2} h_{i,\hat{m}_0[i]\oplus t[i]} \neq \prod_{i=1}^{\ell_2} h_{i,\hat{t}[i]}$ or $\hat{m}_0 \oplus t = \hat{t}$ or $s = \hat{s}$, aborts the experiment, else it returns $(\hat{m}_0 \oplus t, \hat{t})$.

If the experiment returns $\hat{m}_0$ in G_2 and F_3 occurs, then $\mathcal{B}$ returns $x_1, x_2 \in \{0,1\}^{\ell_2}$ s.t. $x_1 \neq x_2$ and $\prod_{i=1}^{\ell_2} \widetilde{g}_{(i,x_1[i])} = \prod_{i=1}^{\ell_2} \widetilde{g}_{(i,x_2[i])}$, which happens with non negligible probability $\epsilon_{\ell_2\text{-col}}$ according to the lemma 4.
Thus, we have:
$\forall \hat{m} \in \mathcal{M}$,

$$|\Pr[\hat{m} \leftarrow G_2] - \Pr[\hat{m} \leftarrow G_3]| \leq \Pr[F_3] \leq \epsilon_{\ell_2\text{-col}}(\lambda).$$

Game G_4: This game is the same as the game G_3 except that aborts and returns 0 on the event $F_4 =$ " The algorithm $\mathcal{A}_1$ returns $(\mathsf{c}, \pi_*, m, \pi)$ and the algorithm $\mathcal{A}_2$ returns $(\hat{m}_0, \hat{\pi})$ such that $\prod_{i=1}^{\ell_1} g_{i,s[i]} = \prod_{i=1}^{\ell_1} g_{i,\hat{s}[i]}$ and $s \neq \hat{s}$", we have:
$\forall \hat{m} \in \mathcal{M}$,

$$|\Pr[\hat{m} \leftarrow G_3] - \Pr[\hat{m} \leftarrow G_4]| \leq \Pr[F_4].$$

We claim that:

$$\Pr[F_4] \leq \epsilon_{\ell_1\text{-col}}(\lambda).$$

We prove this claim by reduction.
Assume the event F_4 happens with non negligible probability, we build the following PPT algorithm $\mathcal{B}$ that play the experiment defined in the lemma 4:
Algorithm $\mathcal{B}\left((\widetilde{g}_{(i,j)})_{i\in[\![\ell_1]\!],j\in\{0,1\}}\right)$**:** simulates the game G_3 to $\mathcal{A}$ except that during the setup generation, for each

$i \in [\![\ell_1]\!]$, it replaces $g_{i,0}$ by $\widetilde{g}_{(i,0)}$, and $g_{i,1}$ by $\widetilde{g}_{(i,1)}$. Runs $(\mathsf{c}, \pi_*, m, \pi) \leftarrow \mathcal{A}_1(\mathsf{set})$, picks $\hat{\theta} \xleftarrow{\$} \Theta$ and runs $(\hat{m}_0, \hat{\pi}) \leftarrow \mathcal{A}_2(\hat{\theta})$. $\mathcal{B}$ uses the extractor on the proof π_*, we note x the witness extracted.

For each $i \in [\![\ell_1]\!]$,

- If $A_{(i,1)} = g_{i,0}^x$, sets $s[i] = 0$.
- Else sets $s[i] = 1$.

$\mathcal{B}$ computes $\prod_{i=1}^{\ell_1} g_{i,s[i]}$ and $\prod_{i=1}^{\ell_1} g_{i,\hat{s}[i]}$, If the event F_4 does not happen, aborts the experiment, else it returns $(s, \hat{s})$.

Thus, we have:
$\forall \hat{m} \in \mathcal{M}$,

$$|\Pr[\hat{m} \leftarrow G_3] - \Pr[\hat{m} \leftarrow G_4]| \leq \Pr[F_4] \leq \epsilon_{\ell_1\text{-col}}(\lambda).$$

If F_1, F_2, F_3, and F_4 does not happen, then the commitment is well formed and the only way to open the commitment is to return $\hat{m} = m$ if $s = \hat{s}$, and to return $t \oplus \hat{t}$ otherwise. We deduce that the game G_4 is the same as the Prob-LDP-Binding experiment in the case where $b = 1$.

Finally, we deduce that:

$$\begin{aligned}
&\forall \hat{m} \in \mathcal{M}, \\
&\Big|\Pr\Big[\hat{m} \leftarrow \mathsf{Exp}^{\mathsf{Prob\text{-}LDP\text{-}Binding}}_{\mathsf{ORRC},\mathsf{LDP},\mathcal{A},0}(\lambda)\Big] - \\
&\Pr\Big[\hat{m} \leftarrow \mathsf{Exp}^{\mathsf{Prob\text{-}LDP\text{-}Binding}}_{\mathsf{ORRC},\mathsf{LDP},\mathcal{A},1}(\lambda)\Big]\Big| \\
&= |\Pr[\hat{m} \leftarrow G_0] - \Pr[\hat{m} \leftarrow G_4]| \\
&\leq 5 \cdot \epsilon_{\mathsf{ext}}(\lambda) + \epsilon_{\ell_1\text{-col}}(\lambda) + 2 \cdot \epsilon_{\ell_2\text{-col}}(\lambda).
\end{aligned}$$

which concludes the proof of the lemma since $5 \cdot \epsilon_{\mathsf{ext}}(\lambda)$, $2 \cdot \epsilon_{\ell_2\text{-col}}(\lambda)$ and $\epsilon_{\ell_1\text{-col}}(\lambda)$ are negligible. $\square$