

Towards Verifiable Differentially-Private Polling

Gonzalo Munilla Garrido*
gonzalo.munilla-garrido@tum.de
Technical University of Munich
Garching, Germany

Matthias Babel
matthias.babel@fim-rc.de
FIM Research Center,
University of Bayreuth
Bayreuth, Germany

Johannes Sedlmeir
johannes.sedlmeir@fim-rc.de
Fraunhofer FIT, Branch Business &
Information Systems Engineering
Bayreuth, Germany

ABSTRACT

Analyses that fulfill differential privacy provide plausible deniability to individuals while allowing analysts to extract insights from data. However, beyond an often acceptable accuracy tradeoff, these statistical disclosure techniques generally inhibit the verifiability of the provided information, as one cannot check the correctness of the participants' truthful information, the differentially private mechanism, or the unbiased random number generation. While related work has already discussed this opportunity, an efficient implementation with a precise bound on errors and corresponding proofs of the differential privacy property is so far missing. In this paper, we follow an approach based on zero-knowledge proofs (ZKPs), in specific succinct non-interactive arguments of knowledge, as a verifiable computation technique to prove the correctness of a differentially private query output. In particular, we ensure the guarantees of differential privacy hold despite the limitations of ZKPs that operate on finite fields and have limited branching capabilities. We demonstrate that our approach has practical performance and discuss how practitioners could employ our primitives to verifiably query individuals' age from their digitally signed ID card in a differentially private manner.

CCS CONCEPTS

• **Information systems** → *Electronic data interchange*; • **Security and privacy** → *Cryptography*; *Human and societal aspects of security and privacy*; **Privacy-preserving protocols**.

KEYWORDS

Digital wallet, exponential noise, privacy, randomized response, SNARK, survey, zero-knowledge proof

ACM Reference Format:

Gonzalo Munilla Garrido, Matthias Babel, and Johannes Sedlmeir. 2022. Towards Verifiable Differentially-Private Polling. In *The 17th International Conference on Availability, Reliability and Security (ARES 2022)*, August 23–26, 2022, Vienna, Austria. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3538969.3538992>

*Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

ARES 2022, August 23–26, 2022, Vienna, Austria

© 2022 Association for Computing Machinery.

ACM ISBN 978-1-4503-9670-7/22/08...\$15.00

<https://doi.org/10.1145/3538969.3538992>

1 INTRODUCTION

Gathering information through polls to produce statistics regarding, e.g., the health, financial status, or demographics of a population bears the risk of exposing individuals' sensitive data during and after the survey. One approach is to anonymize the gathered data centrally, which implies high costs for implementing security measures and still carries ethical risks. Moreover, since interviewees cannot control that their data is adequately anonymized and protected in this paradigm and their level of trust in the surveyor is limited, their response may be subject to bias, specifically with highly sensitive or embarrassing questions.

A simple means to enhance privacy by design and reduce the risk of bias in such polls is through randomized response [57], its variations [6, 25, 30, 41, 58], or more involved forms of local differential privacy (DP) [3, 31–33, 37, 38], which provide plausible deniability by adding noise to interviewees' answers. The noise distribution of these techniques is typically centered around 0 with finite variance [14], so according to the law of large numbers, the mean of the noisy data converges towards the original mean as the sample size increases, improving accuracy. However, in this local approach, there is a lack of response *verifiability* – the interviewer has no assurance that the interviewee, i.e., the adversary in our system, answered truthfully. This lack of verifiability is arguably severe when rewards encourage malicious participation, e.g., there is a monetary incentive to participate but no willingness to answer truthfully.

Verifiable computation can prove the execution of a particular algorithm from truthful inputs without revealing private information [4]. Accordingly, we suggest combining verifiable computation with local differential privacy (LDP) techniques to prove (i) the interviewees' plausible deniability guarantee derived from randomness and (ii) the truthfulness of their deterministic answer, i.e., the value has been signed by a reputed authority. Similar approaches have been discussed, for instance, in [44, 47]. Our approach hence targets polls where there is cryptographic evidence for the answers, e.g., a digital ID card signed by a government, digital diplomas issued by a certified university, or COVID-19 immunity passports certified by pharmacies or doctors. Such attestations are considered, for instance, in the European digital wallet initiative [15, 46, 48]. In this context, it is particularly helpful that the digital certificates involved in the many implementations of digital wallets are in fact anonymous credentials [7, 50, 52], which allows us to extract a user's attribute values without revealing strongly correlating information. Moreover, we believe that in the private sector, attestations derived, for instance, from cryptographically signed statements of bank accounts or insurance claims and their use in verifiable differentially private surveys could have considerable economic

potential, as data markets require technologies that provide verifiability despite privacy protection [20].

There are two main approaches for verifiable computation: trusted execution environments (TEEs) [45], and non-interactive or interactive zero-knowledge proofs (ZKPs) [4, 23, 53]. Given the numerous known vulnerabilities and attacks on TEEs [1, 40, 54] and Intel’s SGX SDK deprecation [35], we decided to focus on ZKP-based approaches. Moreover, non-interactive ZKPs do not require to engage into sequential messaging, so – unlike with interactive ZKP – the prover can convince multiple parties of a claim with a single message [53]. Thus, we opted to use non-interactive ZKPs to enable the verifiability of the computational integrity in the selected DP mechanism.

This paper’s scope covers both binary answers, e.g., “Are you older than 18?”, and numerical answers, e.g., “How old are you?”. We provide plausible deniability for interviewees with differential privacy (DP) mechanisms in the local model, specifically, employing randomized response [57] and exponentially distributed noise [12]. Lastly, we adapt these mechanism such that we can verify their correct execution with ZKPs by employing succinct non-interactive arguments of knowledge (SNARKs) [5, 27], resulting in the primitives represented in Algorithms 1 and 2. We implement the corresponding circuits and evaluate their performance characteristics to assess our approach’s practicality.¹

As randomized response and exponential noise are building blocks for other more complex mechanisms, our scheme could also be extended to prove their verifiability, such as in two-stage randomized response models [30, 41], unrelated question models [25], forced response models [6], LDP models [31–33, 37, 38], private weighted histogram aggregation in crowdsourcing by leveraging multivariate randomized response [58], building histograms [3], or using exponential noise distributions in the central model of DP. Such verifiable forms of DP are also relevant in multilateral protocols that provide economic incentives for participation based on the participants’ contribution. In such settings, one should compute fair rewards from the original data without noise, requiring that the computation of both their deterministic contribution and the shared noisy value is verifiable. An example for such a scenario is fair blockchain-based federated learning, studied by Rückel et al. [47].

This paper is structured as follows. We provide preliminaries in Section 2, discuss the SNARK-based approach and its implementation in Section 3, and evaluate it in Section 4. Lastly, we comment on related work in Section 5, discuss our approach in Section 6, and conclude the paper in Section 7.

2 PRELIMINARIES

2.1 Differential Privacy

We consider a collection of records from a population (dataset) D to belong to the universe of possible datasets \mathcal{D} . We let $D' \sim D$ denote neighboring datasets, i.e., D and D' differ by only one record. Differential privacy, introduced by Dwork et al. in 2006 [13], formalizes a mathematical definition of privacy whereby an analysis’ output distribution is nearly the same across all neighboring datasets. The

indistinguishability between datasets is parameterized by $\epsilon > 0$. The higher ϵ , the easier it is to identify datasets.

DEFINITION 1. ((ϵ, δ) -Differential Privacy [14]). A randomized mechanism \mathcal{M} is (ϵ, δ) -differentially private iff for any neighboring dataset $D' \sim D$, and any set of possible outputs $\mathcal{S} \subseteq \text{Range}(\mathcal{M})$,

$$\Pr[\mathcal{M}(D) \in \mathcal{S}] \leq e^\epsilon \cdot \Pr[\mathcal{M}(D') \in \mathcal{S}] + \delta.$$

Having a non-zero δ relaxes the strict ϵ bound for possible but unlikely events; this type of guarantee is called *approximate* DP, whereas with $\delta = 0$, we obtain *pure* DP. A randomized mechanism \mathcal{M} typically ensures DP by adding carefully calibrated random noise to the output of a deterministic function $f(\cdot)$, for example, by adding exponentially distributed noise [12] to a count, average, or median. Furthermore, another factor beyond ϵ that calibrates noise is the sensitivity of $f(\cdot)$, which measures the maximum variation of the output as the input dataset D changes (denoted as Δ).

Lastly, it is important to note that a DP mechanism \mathcal{M} follows *sequential composition* [14], i.e., if \mathcal{M} is computed n times over a dataset \mathcal{D} with ϵ_i , in effect, the total ϵ is given by $\sum \epsilon_i$. Thus, the results become less private with every query. Yet, a system can effectively impede an attacker from averaging out the noise through a sequence of DP results by blocking subsequent queries or deterministically generating the randomness based on the query parameters.

2.2 Local Differential Privacy

Definition 1 corresponds to the *central model*, in which a trusted curator collects data points and adds noise from a distribution whose variance is tuned by the function’s sensitivity (Δ) and the required degree of plausible deniability (ϵ). In this paper, we focus on the *local model*, whereby the data subject obfuscates the data points directly before sharing. While the local model typically provides less accuracy than the central model, the data subject does not need to trust the curator from a privacy perspective.

For the local setting, we adopt similar notation to [22, 36]. We let \mathcal{X} contain all the possible records of a population, where $x \in \mathcal{X}$ holds a particular individual’s data. We let $f : \mathcal{X} \rightarrow [l, u]$ be a function that maps each element of $x \in \mathcal{X}$ to $f(x) \in [l, u]$, where $[l, u]$ is the set of integers between the lower bound $l \in \mathbb{N}$ and the upper bound $u \in \mathbb{N}$. In practice, $f(\cdot)$ provides information about an individual, e.g., their age, income, height, or blood pressure. Let $\mathcal{M} : \mathcal{X} \rightarrow [0, n]$ denote a randomized mechanism that maps each deterministic query output $f(x) = i \in [l, u]$ to each possible value $j \in [l, u]$ following a probability distribution that depends on the value i . In this setting, a randomized mechanism \mathcal{M} provides local (ϵ, δ) -DP iff for every pair of inputs $x, x' \in \mathcal{X}$, and for every possible output $j \in [l, u]$:

$$\Pr[\mathcal{M}(x, f) = j] \leq e^\epsilon \cdot \Pr[\mathcal{M}(x', f) = j] + \delta.$$

We will cover two mechanisms that satisfy local DP: randomized response [57] for binary data and exponentially distributed noise [12] for numerical data.

Randomized response. Warner [57] introduced randomized response in 1965 to provide plausible deniability to interviewees, which encouraged them to answer truthfully, thus reducing bias in surveys. The interviewees would answer queries $f(\cdot)$ of the

¹The source code can be found at <https://github.com/applied-crypto/DPfeatZKP>.

form “Are you a [...]?”. The following algorithm is well-known to be $(\ln 3, 0)$ -differentially private [14]:

- (1) Flip a coin.
- (2) If heads, answer truthfully.
- (3) Else, flip the coin again and answer “Yes” if heads and “No” otherwise.

Exponentially distributed noise. Early work by Dwork et al. [12] shows that noise distributed as $\Pr[x] \propto \exp(-\frac{\epsilon|x|}{\Delta})$ fulfills $(\epsilon, 0)$ -DP. We leverage exponentially distributed noise for locally obfuscating numerical data to answer queries $f(\cdot)$ of the form “How many [...]?”, i.e., count queries, by adding noise to the deterministic output of $f(\cdot)$ in this manner: $\mathcal{M}(x, f) = f(x) + \mathbf{noise}$. As the responses are local, we must ensure indistinguishability between any $f(x)$ and $f(x')$; thus, we must set Δ according to the output range of $f(\cdot)$. In practice, to ensure that an attacker cannot easily distinguish individuals in the extreme-case scenario, e.g., a newborn from a 128-year old person, with the query “How old are you?” we set $\Delta = |\max_x f(x) - \min_x f(x)| = |128 - 0|$.

As presented above, both mechanisms fulfill pure DP; however, generating exponentially distributed noise is subject to approximation errors in practical implementations and, thus, we only achieve approximate DP instead [21]. We shed more light on this issue in Section 3.

2.3 Proof Systems and SNARKs

There have been several key milestones in the work towards cryptographically verifiable computations. Babai [2] studied interactive proofs between a *prover* and a *verifier* and analyzed which problems can be checked by a polynomially bounded verifier when adding randomness and interaction. Fiat and Shamir [17] then introduced a heuristic to replace the verifier with a random oracle, which one can implement with a secure hash function. Nonetheless, one uses the word “argument” instead of “proof” in this case because the existence of a secure hash function has not been proven mathematically so far and is rather a working hypothesis, also bound to today’s compute and (differential) cryptanalysis capabilities. Goldwasser et al. [23] proved that one could verify a large class of problems probabilistically in this way, where the verifier additionally does not even need to learn anything beyond the statement’s correctness. While practical applications of the accordingly termed ZKPs were rare after these early developments, a period of rapid improvements started in the mid-2000s and led to the computationally efficient (quasi-linear complexity) generation of succinct arguments of logarithmic size and verification time, called SNARKs [5, 27].

The research community has since developed other flavors such as scalable transparent arguments of knowledge (STARKs), which differ in the setup procedure, proof size, and cryptographic assumptions, but have similar functional aspects. These new approaches allow succinct or scalable zero-knowledge proofs for the correctness of arbitrary statements and, thus, practical verifiable computation, i.e., proofs for the correct execution of a program without displaying all inputs, outputs, or intermediate steps. Additionally, one can reveal Merkle proofs or other cryptographic relations like the public keys corresponding to a digital signatures by a reputed institution on the inputs to force the prover to use unknown but fixed variables. Arguably, the core area of application of ZKPs today is in distributed

Table 1: Notation

\mathcal{X}	Universe of records
$x \in \mathcal{X}$	Individual record
$f : \mathcal{X} \rightarrow [l, u]$	Query function
l	Lower bound, $\min_x f(x)$
u	Upper bound, $\max_x f(x)$
$\mathcal{M} : [l, u] \rightarrow [l, u]$	Randomized mechanism
Δ	Sensitivity of $f(\cdot)$, $ u - l $
ℓ	Noise added to $f(\cdot)$
nBits	Number of bits representing ℓ
p_k	Bias of bit k , $\Pr[\ell_k = 1]$
d	Precision

ledgers, where because of redundant execution, cheap (succinct) verification without revealing sensitive data is important to solve scalability issues and mitigate excessive data visibility [4, 51].

3 IMPLEMENTATION

In this section, we first describe how to adapt standard implementations of uniform randomness generation, randomized response, and exponentially distributed noise (see Section 2) such that ZKPs can verify their use. For our implementation, we employ Circom, a well-known, open-source technology stack for implementing ZKPs [34]. Circom is a domain specific programming language and compiler that translates JavaScript-like arithmetic circuits in a rank one constraint system (R1CS), on behalf of which further libraries (e.g., SnarkJS) can generate SNARKs. A circom-specific variable type to explicitly define constraints is called *Signal*. The programming of these signals is restricted by the underlying quadratic arithmetic program (QAP), which the R1CS encodes, to use only quadratic constraints inside one Signal. Therefore, a Signal can only be assigned once and is immutable. For this reason, calculations often have to be split into multiple sub-calculations. Moreover, branchings and loops can only be used in a restricted way, for instance, the maximum number of iterations must be specified by a constant instead of a variable or Signal.

In what follows, we will use the roles prover for the survey participant and verifier for the surveyor, and the notation specified in Table 1. We also assume that both of them have a dedicated key-pair that they can use for end-to-end encrypted, authenticated communication and for recognizing each other, or another means to bind them to a specific secret key, for instance, through an anonymous credential with private holder binding [7]. Such anonymous credentials can be implemented with specific-purpose ZKPs [7] and with SNARKs [11, 49]. Key-pairs are a common way to facilitate the generation of verifiable randomness, for instance, in Algorand’s consensus mechanism [10].

3.1 Verifiable Uniform Randomness

To achieve uniform randomness that cannot be spoiled unilaterally by either the prover or the verifier, we employ two inputs and a hash function as a random oracle [8]. More specifically, we sign a challenge that the verifier specifies with the prover’s private

Algorithm 1: Verifiable randomized response for binary data and uniform randomness (“unbiased coins”).

Data: v : binary truthful value (“Yes” or “No”); a : prover contribution to randomness (secret key); b : verifier contribution to randomness (challenge).

Result: Differentially private answer.

```

1 Function VerifiableUnifRand( $a, b$ ):
2    $s = \text{sign}(a, b)$  // sign challenge with secret key
3    $r = \text{hash}(s)$  //  $r$  is an array of bits
4   return  $r$ 
5 Function VerifiableRandomizedResponse( $v, a, b$ ):
6    $r = \text{VerifiableUnifRand}(a, b)$ 
7   if  $r[0] = 0$  then
8     return  $v$ 
9   else if  $r[1] = 0$  then
10    return No
11  else
12    return Yes

```

key and hash the result. As the private key is determined by the fixed prover’s public key, neither of the two parties can bias the resulting randomness without collusion. We use Poseidon² – a relatively new hashing algorithm that was specifically developed for use in ZKPs and that is already being used in many blockchain-based applications on Ethereum and, therefore, to some extent battle-tested [24]. We represent this building block as a function in Algorithm 1 between lines 1 and 4, using existing components in Circom for EdDSA signature verification, Poseidon, and conversion of (large) integers to binary representation. Assuming that the Poseidon hash function is a random oracle and the keypair was created without anticipating the survey and the verifier’s challenge, this gives us an array of 254 unbiased random bits.

3.2 Verifiable Randomized Response

Randomized response is simple to verify with ZKPs by utilizing the verifiable uniform randomness function (see Algorithm 1). In practice, without loss of generality, we only consider the least two significant bits of the random number generated. For the randomized response algorithm presented in Section 2 and presented formally in Algorithm 1, we need to sample at least once (last bit) and at most twice (second-last bit), depending on the first coin flip. The source code from Fig. 1 implements this in Circom. As if-statements are not natively possible in R1CS and, therefore, only available with restrictions in Circom, we arithmetize the corresponding statements in lines 7 to 12 from Algorithm 1 into the lines 39 to 40 from Fig. 1.

3.3 Verifiable Exponentially Distributed Noise

The exponentially distributed noise adaptation to ZKPs is not as straightforward as with randomized response because it typically involves floating point operations and rounding. After trying different implementations of exponentially distributed noise generation – we briefly cover the journey in Section 6 – we successfully adapted the method proposed by Dwork et al. [12] to ZKP, which we present in Algorithm 2: In their method, Dwork et al. approximated exponentially distributed noise of $\Pr[x] \propto \exp(-\frac{\epsilon|x|}{\Delta})$ with the Poisson

²Using other hashing mechanisms is possible, yet the performance can become considerably worse – for instance, in the case of SHA256, around 30x.

```

1 pragma circom 2.0.0;
2
3 include "./poseidon.circom"; // Poseidon hashing
4 include "./bitify.circom"; // Bit array conversion
5 include "./eddsaposeidon.circom"; // Signature checking
6
7 template Main() {
8   signal input value; // v
9   signal input challenge;
10  signal input R8[2]; // elliptic curve element of
11  // signature
12  signal input S; // field element of signature
13  signal input pk[2]; // public key
14
15  // check signature on challenge against public key
16  component eddsaVerifier = EdDSAPoseidonVerifier();
17  eddsaVerifier.Ax <== pk[0];
18  eddsaVerifier.Ay <== pk[1];
19  eddsaVerifier.S <== S;
20  eddsaVerifier.R8x <== R8[0];
21  eddsaVerifier.R8y <== R8[1];
22  eddsaVerifier.M <== challenge;
23  eddsaVerifier.enabled <== 1; // checks signature implicitly
24
25  // hash signature and convert this randomness to bit array
26  component hash = Poseidon(3);
27  component bitify = Num2BitsStrict();
28  hash.inputs[0] <== R8[0];
29  hash.inputs[1] <== R8[1];
30  hash.inputs[2] <== S;
31  bitify.in <== hash.out;
32  signal randSeq[254];
33  for(var i = 0; i < 254; i++) {
34    randSeq[i] <== bitify.out[i];
35  }
36
37  // determine result from randomness
38  signal rand;
39  signal output out;
40  rand <== randSeq[0] * randSeq[1];
41  out <== (1 - randSeq[0]) * value + rand;
42 }
43
44 component main {public [challenge, pk]} = Main();

```

Figure 1: Circom code for a component that implements verifiable randomized response.

distribution, fulfilling (ϵ, δ) -DP. Their method samples noise by producing a sequence of biased bits equal in number to the number of bits in the binary expansion of the noise ℓ . The algorithm flips an extra bit to add a sign $(\pm\ell)$. The bias of each bit $k \in \{0, \dots, n\text{Bits}\}$ representing ℓ in binary is given in Section 4.1 of [12] by

$$\Pr[\ell_k] := \Pr[\ell_k = 1] = \left(1 + \exp\left(\frac{\epsilon \cdot 2^k}{\Delta}\right)\right)^{-1}.$$

To generate biased bits from unbiased bits, we include in Algorithm 2 a well-known technique: first, we expand in binary the bias p_k of a bit k . Afterward, the algorithm sequentially examines random unbiased bits until one differs from the corresponding bit in the binary expansion of p_k and, subsequently, outputs the complement of the random unbiased bit [12]. Essentially, this approach allows to simulate biased coins up to a pre-defined precision with unbiased coins. However, the method employed has three limitations.

The *first* limitation entails several issues that relate to representing with a limited precision d the bias of the bits composing p_k ,

i.e., d is the number of bits available for representation. Nonetheless, the probability of the inner loop not terminating for $j < d$ and, therefore, raising an error decays with 2^{-d} . Thus, we can easily choose d such that the likelihood of this event is negligible (line 17 of Algorithm 2). Furthermore, we show that we can provide enough precision in our circuit: The randomness generated from a single Poseidon hash could provide a precision of around $2^{252} \approx 10^{75}$, i.e., $d \approx 75$. By using multiple rounds of hashing and signing, we could also generate more random bits and account for higher precision needs. Additionally, we restrict noise values ℓ to the interval $[l, u]$, where u and l are the deterministic function's output upper and lower bounds, respectively. For our experiment on polling individuals' age, we employed the algorithm with $d = 20$ and $\Delta = |u - l| = 128$. These example values require the algorithm to represent ℓ with $n\text{Bits} = 7$ bits and, in turn, generate one instance of noise with $d \cdot 7 + 1 = 141 < 256$ bits ("times d " because each of the 7 bits' bias will be expanded to d bits and one more for the sign). In other words, a single round of hashing and signature verification is sufficient (and would still be sufficient for $d = 35$, which corresponds to an error bound of $2^{-35} \approx 10^{-10}$ when approximating probabilities [12]), and a negligible probability to raise an error (upper bound $7 \cdot 2^{-20}$). We could also aim for the typical machine accuracy of 10^{-16} by using $d > 16 \cdot \log_2(10) \approx 53.1$, i.e., $d \geq 54$, which would involve the creation of two independent random bit arrays.

These design decisions allow us to approximate the Poisson distribution with an error bound that we can determine and control ex-ante, when designing the survey. Thus, we achieve (ϵ, δ) -DP with a statistical difference of $\delta = n\text{Bits} \cdot 2^{-d} = 7 \cdot 2^{-20}$ [12]. Consequently, for improving the DP guarantee on ϵ , we only need to increase d . Moreover, the probability mass outside the considered interval $[-2^d, 2^d]$ is redistributed inside the interval, leading to an additional statistical difference of $2 \exp(-(\epsilon \cdot 2^d)/\Delta)$ that we let the term $n\text{Bits}$ absorb [12].

The *second* limitation is the zero probability assigned to noise values of a binary expansions with more bits than $n\text{Bits}$ (i.e., noise outside of $[l, u]$). Dwork et al. proposed to constrain the algorithm's output, i.e., deterministic answer + **noise**, to $n\text{Bits}$ and return the deterministic answer in case there is an overflow. According to Dwork et al. [12], as the distribution in the range $[l, u]$ is exponential, we maintain the same privacy guarantee by increasing the probability of not adding noise by a *trivial* amount (i.e., δ increases). We execute a modulo operation to remap any output value outside $[l, u]$ back in that range to reduce such an increase (lines 21 and 23 of Algorithm 2). Intuitively, a modulo operation on the output preserves DP as it is a post-processing step and, also, will re-distribute the outputs in the range instead of on one value. Formally, the proof may be found in Lemma 3 of Wang et al. [56].

The *third* limitation comes with flipping an unbiased bit to assign the sign of the noise, which converts a Poisson distribution into a two-sided distribution with double the probability on its center, i.e., of noise = 0. While Dwork et al. did not address this issue in [12], we could follow the approach of Champion et al. [9] of rejecting -0 and executing the algorithm again (section 3.3 of [9]). DP is maintained as the number of failures is independent of the noise. However, instead, to remain computationally performant,

we output a uniformly sampled value within $[l, u]$ if -0 (lines 19 to 23 of Algorithm 2), effectively removing the excess probability at 0. Intuitively, we preserve DP by adding more noise to the output distribution. Formally, we provide this justification: Let P_{old} be some DP distribution on N discrete values and P_{new} such that $P_{\text{new}}(x) = \alpha P_{\text{old}}(x) + (1 - \alpha)/N$. Obviously, this is a probability distribution. In our case, $1 - \alpha$ is the probability of obtaining noise -0 . For any D and D' that differ in at most one record,

$$\begin{aligned} \Pr[\mathcal{M}_{\text{new}}(D) \in S] &= \alpha \cdot \Pr[\mathcal{M}_{\text{old}}(D) \in S] + (1 - \alpha) \cdot \frac{|D|}{N} \\ &\leq \alpha e^\epsilon \cdot \Pr[\mathcal{M}_{\text{old}}(D') \in S] + \alpha \delta + (1 - \alpha) \cdot \frac{|D|}{N} \\ &= \alpha e^\epsilon \cdot \left(\frac{\Pr[\mathcal{M}_{\text{new}}(D') \in S]}{\alpha} - \frac{|D'| \cdot (1 - \alpha)}{N\alpha} \right) + \alpha \delta + (1 - \alpha) \cdot \frac{|D|}{N} \\ &= e^\epsilon \cdot \Pr[\mathcal{M}_{\text{new}}(D') \in S] + \alpha \delta + (1 - \alpha) \cdot \frac{|D| - e^\epsilon |D'|}{N} \\ &\leq e^\epsilon \cdot \Pr[\mathcal{M}_{\text{new}}(D') \in S] + \alpha \delta + (1 - \alpha) \cdot \frac{1}{N}. \end{aligned}$$

Given that $\epsilon \geq 0$, then $e^\epsilon \geq 1$. Moreover, $|D| - |D'| \leq 1$ because they are neighboring. Thus, since we choose to re-distribute the excess weight for noise -0 ($\alpha = 1 - \frac{1}{2}\text{Prob}(0)$), δ may grow to at most $\frac{1}{2}\text{Prob}(0) \cdot \frac{1}{N}$. The above formulation is a universal upper bound: Essentially, it proves that the convex combination of an (ϵ, δ_1) mechanism and a uniform distribution with pointwise weight $\delta_2 = \frac{1}{N}$ (which is obviously $(0, \delta_2)$ -DP) is (ϵ, δ) -DP, where δ is the convex combination of δ_1 and δ_2 . Future work could focus on obtaining a tighter δ bound specifically for the Poisson distribution or employ the method from Champion et al. [9], which would maintain a δ upper bound of $n\text{Bits} \cdot 2^{-d}$. Altogether, this approximation allows us to sample exponentially distributed noise preserving (ϵ, δ) -DP in a way that we can successfully verify with ZKPs. We depict an example of the resulting output distribution in Fig. 2.

As Circom does not allow for branching, i.e., implementing conditional checks and breaking or continuing loops, besides the workaround for if-statements, we had to introduce some additional Signals (see Fig. 3). These Signals allow us to determine the correct return value although all iterations from the loop are simulated in the circuit. We did this by introducing Signal arrays `hit1...nBits` with length d that help to identify the firsts unequal pair of bits in one (j) loop, where the loop would break in Algorithm 2. This approach ensures that only the first occurrence of unequal bits (index i) is taken into account for the calculation of the biased randomness. Moreover, we multiply the inverted binary result from `isEqual`, which compares the corresponding bits of the random sequence r_j and the probability B_{j,p_k} , with the bit of the probability and the hit bit-value, which is 1 as long as the result of `isEqual` of the last iteration of the loop was 1, essentially $[1_0, \dots, 1_{i-1}, 1_i, 0_{i+1}, \dots, 0_{d-1}]$. Therefore, only at the first inequality of those two bits the probability bit is not multiplied with zero, and, thus, can be taken into account for the noise. In other words, `eval3[k][j]` is the "running return value" after the $j + 1$ st iteration of the loop, and is set to 1 only if the j th bit of the probability is one, the j th bit of the random sequence is 0, and the first time that the probability and random bit array are different occurs at position j as well.

Algorithm 2: Verifiable exponentially distributed noise generation for numerical data. By $x \bmod(l, u)$ we denote $l + (x \bmod(u - l))$.

Data: v : integer-valued truthful value; u : upper bound; l : lower bound; $\Delta = |u - l| \geq 0$: sensitivity of query function; $\epsilon \geq 0$: privacy parameter; $d \geq 0$: precision of binary expansion; a : prover contribution to randomness (secret key); b : verifier contribution to randomness (challenge).

Result: $v + \text{noise} \sim \text{Pois}(v | \frac{\epsilon}{\Delta})$.

```

1 Function VerifiableExponentialNoise( $v, \Delta, \epsilon, d$ ):
2    $B_K = \text{BinaryExpansion}(\Delta)$ 
3    $B_v = \text{BinaryExpansion}(v)$ 
4    $B_r = []$  //  $B_r$  stacks biased bits
5   for  $k \leftarrow 0$  to  $\text{NumBits}(B_K)$  do
6      $p_k = \frac{1}{1 + \exp(2^k \frac{\epsilon}{\Delta})}$ 
7      $B_{p_k} = \text{BinaryExpansion}(p_k)$ 
8     //  $r$  has at least  $d$  bits
9      $r = \text{VerifiableUnifRand}(a, b)$ 
10    for  $j \leftarrow 0$  to  $d$  do
11      // Where  $d$  is the least significant bit
12       $r_j = r[j]$  //  $r_j \in \{0, 1\}$ 
13      if  $r_j = B_{j,p_k}$  then
14        continue
15      else
16         $B_r.\text{push}(B_{j,p_k})$ 
17        break
18      if  $j = d$  then
19        return RaiseError
20
21   $\text{noise} = \text{DecimalExpansion}(B_r)$ 
22   $\text{sign} = \text{VerifiableUnifRand}(a, b)[0]$ 
23  if ( $\text{noise} = 0$  and  $\text{sign} = 0$ ) then
24    return
25     $\text{DecimalExpansion}(\text{VerifiableUnifRand}(a, b) \bmod(l, u))$ 
26  else
27    return  $[(v + (2 \cdot \text{sign} - 1) \cdot \text{noise}) \bmod(l, u)]$ 

```

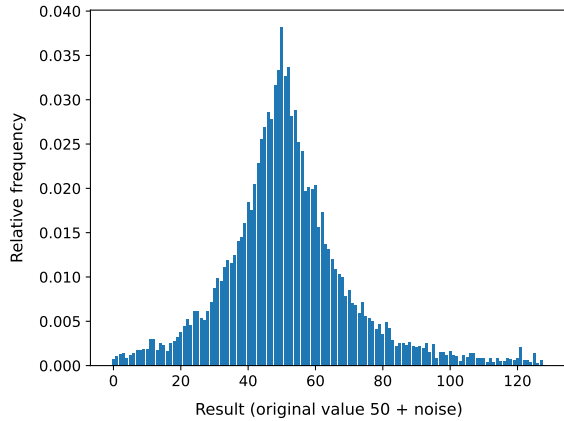


Figure 2: Example histogram for $l = 0$, $u = 128$, $d = 20$, $\epsilon = 10$, and true value $v = 50$ with a sample size of 10000.

```

1
2 // include statements as before, plus modulo component
3 template Main( $n\text{Bits}, d$ ) {
4   signal input challenge, value
5   signal input prob[ $n\text{Bits}$ ][ $d$ ]; // binary expansions of  $p_k$ 
6   signal input R8[2], S, pk[2]; // signature and public key
7   // check the EdDSA signature of the challenge against pk and
8   // put it in the hash component to create randSeq, as in
9   // Figure 1 (lines 10 to 34).
10  ...
11  ...
12  component isEqual[ $n\text{Bits}$ ][ $d$ ];
13  signal noiseBits[ $n\text{Bits}$ ];
14  signal eval1[ $n\text{Bits}$ ][ $d$ ];
15  signal eval2[ $n\text{Bits}$ ][ $d$ ];
16  signal eval3[ $n\text{Bits}$ ][ $d + 1$ ];
17  signal hit[ $n\text{Bits}$ ][ $d + 1$ ];
18
19  // run the algorithm to create biased coins
20  for ( $\text{var } i = 0; i < n\text{Bits}; i++$ ) {
21    for ( $\text{var } j = 0; j < d; j++$ ) {
22      isEqual[ $i$ ][ $j$ ] = IsEqual();
23    }
24  }
25  for ( $\text{var } k = 0; k < n\text{Bits}; k++$ ) {
26    hit[ $k$ ][0] <== 1;
27    eval3[ $k$ ][0] <== 0;
28    for ( $\text{var } j = 0; j < d; j++$ ) {
29      isEqual[ $k$ ][ $j$ ].in[0] <== prob[ $k$ ][ $j$ ];
30      isEqual[ $k$ ][ $j$ ].in[1] <== randSeq[ $k * d + j$ ];
31      hit[ $k$ ][ $j + 1$ ] <==
32      hit[ $k$ ][ $j$ ] * isEqual[ $k$ ][ $j$ ].out;
33      eval1[ $k$ ][ $j$ ] <== hit[ $k$ ][ $j$ ] * (1 - isEqual[ $k$ ][ $j$ ].out);
34      eval2[ $k$ ][ $j$ ] <== eval1[ $k$ ][ $j$ ] * prob[ $k$ ][ $j$ ];
35      eval3[ $k$ ][ $j + 1$ ] <== eval3[ $k$ ][ $j$ ] + eval2[ $k$ ][ $j$ ];
36    }
37    noiseBits[ $k$ ] <== eval3[ $k$ ][ $d$ ];
38  }
39
40  component numify[2];
41  // compute exponential noise from its binary representation
42  numify[0] = Bits2Num( $n\text{Bits}$ );
43  for ( $\text{var } i = 0; i < n\text{Bits}; i++$ ) {
44    numify[0].in[ $i$ ] <== noiseBits[ $i$ ];
45  }
46  signal absNoise <== numify[0].out;
47  signal positiveNoise <== randSeq[ $n\text{Bits} * (d + 3)$ ] * (value +
48  absNoise);
49  signal noisedResult <== (1 - randSeq[ $n\text{Bits} * (d + 3)$ ]) * (value
50  - absNoise) + positiveNoise;
51
52  // generate uniformly distributed noise
53  numify[1] = Bits2Num( $n\text{Bits}$ );
54  for ( $\text{var } i = 0; i < n\text{Bits}; i++$ ) {
55    numify[1].in[ $i$ ] <== randSeq[ $((d + 2) * n\text{Bits}) + i$ ];
56  }
57
58  component isZero = IsZero(); // check if noise == -0
59  isZero.in <== absNoise;
60  signal isUnif <== isZero.out * (1 - randSeq[ $n\text{Bits} * (d + 3)$ ]);
61  signal unif <== isUnif * numify[1].out;
62  signal result <== (1 - isUnif) * noisedResult + unif;
63
64  component modulo = Modulo();
65  modulo.in <== result;
66  modulo.mod <== 128;
67  signal output out <== modulo.out;
68 }
69
70 component main {public [ $\text{challenge}, \text{pk}$ ]} = Main(7, 22);

```

Figure 3: Circom code for generating verifiable LDP noise with $l = 0$ and $u = 128$. Import statements, signal definitions, and EdDSA verification omitted (see also Figure 1).

3.4 Application: Verifiable Differentially-Private Polling with Anonymous Credentials

With the primitives presented in Algorithms 1 and 2, and an implementation of anonymous credentials with Circom, we can now implement verifiable, differentially private polling. Note that a Circom-based implementation of anonymous credentials allows to selectively disclose attributes from a digital certificate that corresponds to a Merkle tree with a signed root, and incorporate authenticity checks, private holder binding, expiration, revocation, and predicate proofs such as range proofs. We first sketch an example of a hypothetical setting. Digitally signed attestations of a person’s attributes are stored in their “digital wallet” – a mobile application – in the form of an anonymous credential, which could contain personal information such as the holder’s name, age, and gender, as well as a digital signature from an institution that the surveyor trusts regarding the authenticity of the information, e.g., a government or hospital. The digital wallet can respond to so-called proof requests [50] that include requirements from the verifier’s side what the survey participant should prove. In our case, this could include the following requirements:

- Prove knowledge of (i) an authentic anonymous credential, issued by some institution, and (ii) knowledge of the secret key associated with the public key for the private holder, which is a binding included as one of the attributes in the anonymous credential.
- Prove that the anonymous credential is (i) not expired (range proof on expiration attribute) and (ii) not revoked (proof about set-inclusion or exclusion, referring to some public accumulator value as specified by the verifier).
- Reveal the result of our implementation of verifiable randomized response or exponentially distributed noise, applied to one of the (boolean or integer-valued) attributes in the credential. The attribute is represented by the issuer’s signature on the anonymous credential, for instance, the attribute could be a leaf in a Merkle tree whose root is signed by the issuer.

In the case of a SNARK, the wallet (or the proof request) would also need to contain the structured reference string (proving key) generated in a setup procedure. It is important that while generally this proving key must be generated in a multi-party computation, in this case, it can be generated by the surveyor alone: Any party that knows how the structured reference string was created can fake proofs, but the privacy guarantees are not harmed in this case [18].

When the surveyor does not leak the “toxic waste” used for creating the structured reference string, the ZKP’s soundness guarantee provides a chain of trust for the attribute, which is not directly revealed but modified through verifiable noise. Lastly, the verifier (surveyor) can cryptographically check that the attribute and the survey participant’s secret key for private holder binding are used as private inputs for the LDP mechanism, and that the challenge as specified by the surveyor is used. We illustrate the survey process with anonymous credentials and verifiable differential privacy in Fig. 4.

4 EVALUATION

In this section, we discuss the performance and practicality of our approach for verifiable LDP. We restrict the discussion to verifiable exponentially distributed noise because it includes strictly more complex operations, so by demonstrating its reasonable performance, we can conclude that verifiable randomized response also is practical. The implementation process of our verifiable LDP approach with exponentially distributed noise was two-tiered. First, we implemented Algorithm 2 in Javascript and verified that it indeed yields an exponential distribution, where values that – after adding the LDP noise – exceed the range of the output are instead displayed without added noise. Secondly, we employed the Poseidon hash function to create a random Oracle that generates verifiable randomness jointly from the prover’s and the verifier’s input. Next, we implemented the corresponding circuits in Circom.

Our choice of Algorithm 2 and our choice of implementation as displayed in Fig. 3 yielded a highly efficient implementation for creating verifiable, Poisson-distributed LDP: Using the Poseidon hash function, our circuit has 5997 R1CS constraints. On an Ubuntu 20.04 virtual machine with 4 virtual cores that runs on a commercial standard Laptop (Dell Latitude 7400 with an Intel i7 8665U CPU), proof creation – which is typically the bottleneck for using ZKPs – takes around 2.2 s when using the Groth16 proof system [26] on the Barreto-Naehrig curve over a 254 bit prime field (bn128), Web assembly for witness generation, and Javascript for proof generation. The size of the proving and verification key are around 3.4 MB and 3.5 kB, respectively. These sizes suggest that proof generation would also be practical on a web-based mobile application, although future research should validate this assumption.

With an optimized tool, performance is even better: Using an optimized C++-based witness generation and a proof generation based on x86 Assembly for Intel processors (“Rapidsnark”, see [28]), proof generation is reduced to only 140 ms. Because to date there is no available optimized tool for proof verification, this operation still takes around 0.8 s in Javascript. Proof verification in a complex, combined survey may even reduce complexity on the surveyor’s side because the complexity of SNARK verification is not dependent on the complexity of the original computation for which the survey participant proves integrity. Moreover, we tested the deployment of a smart contract verifier on Ethereum, which could be used for blockchain-based, incentivized differentially private surveys and, therefore, general data protection regulation (GDPR)-compliant applications on personal data. We measured the smart contract’s deployment cost at around 1, 150, 000 gas and its invocation at around 300, 000 gas.

5 RELATED WORK

While there is current and extensive research in the local model of DP [3, 31–33, 37, 38] and in ZKPs [4, 5, 27], there are only few publications that bridge both technologies.

We identified four studies that are close to ours. Rückel et al. [47] propose an architecture to share weights from federated learning models in a verifiable DP manner and add verifiable noise to the private weights. However, their approach does not acknowledge that their discretization of the Laplace distribution only fulfills *approximate* DP, and does not propose a bound for δ or an approach that

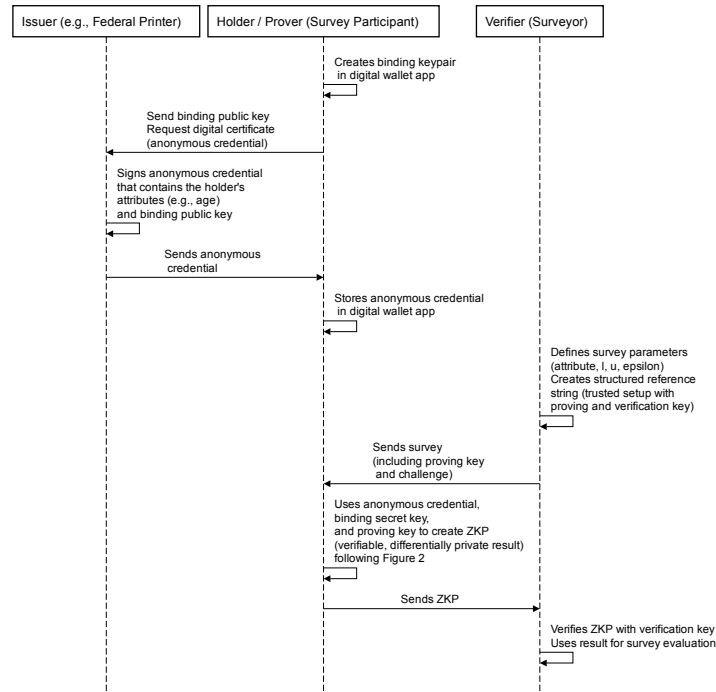


Figure 4: Process of participating in a survey with verifiable differential privacy.

works for high precision requirements, as they use the approximate inverse cumulative distribution function (CDF) as input for the circuit. Moreover, Tsaloli et al. [55] only provide a high-level motivation for using ZKPs for verifiable differential privacy, without implementation details. Furthermore, while Kato et al. [39] provide details on how to create fair randomness with a related technology (secure-multiparty computation), they do not attempt to make the result of the algorithm verifiable, i.e., they cannot provide a cryptographic check of the truthful value from, e.g., an anonymous credential, before adding LDP noise.

Lastly, Narayan et al. [44] discuss the opportunities of verifiable differential privacy, yet they provide no details of their ZKP-based implementation. For instance, they do not elaborate how they consider rounding and how they achieve guarantees on the accuracy of their verifiable *pure* DP proposal. Additionally, their approach focuses on the *central* model of DP instead and shows impractical performance, as it requires 2 hours of proof generation for 32 servers. Nonetheless, when implemented with more recent ZKP libraries, their performance may be closer to ours as advances in proving times over the last years have been dramatic.

6 DISCUSSION

While adapting randomized response to ZKPs is straightforward, we investigated several approaches before successfully implementing exponentially distributed noise with ZKPs. This section discusses the process we followed to arrive to the implementation described in Section 3.

Adapting a DP mechanism that leverages exponential noise to ZKP has two significant challenges. Conceptually, since ZKPs can

verify an arbitrary program, sampling from an exponential distribution may seem straightforward. Unfortunately, in practice, repeated operations with floats that involve rounding in classical software are challenging to implement because the range of numbers in the nominator and denominator is bounded by a large prime, and repeated rounding is costly since the complexity of the ZKP always needs to account for the worst possible case. Furthermore, the propagation of the corresponding errors becomes challenging to control. Thus, the generality of computations that ZKPs can cover well is initially limited to arithmetic operations on prime fields and their corresponding primitives such as hash functions and signatures.

The second major challenge is the inability of finite computers to fulfill the definition of DP on the real line. Mironov [43] was the first to demonstrate that implementing a DP mechanism with the floating-point arithmetic of finite computers does not guarantee DP. Mironov proposed to solve this issue while maintaining ϵ -DP with the Snapping mechanism [43], and recently, Naoise et al. proposed secure random sampling [29]. However, while their output noise is discrete, we must still handle floats that ZKPs cannot process efficiently.

Therefore, we first thought of discretizing the support of the Laplace distribution (well-known to be ϵ -DP [14]) by sampling from its inverse CDF with a finite input range $\{1, \dots, d\}$, where d denotes the precision Circom [34] can handle – similarly to Rückel et al. [47]. However, we were not able to determine provable guarantees on δ for the approximate DP mechanism. Thus, we turned to the *Stone-Weierstrass theorem* to approximate a polynomial so close to the Laplacian probability density function (PDF) that the approximation error would be negligible. Furthermore, because the approximated

PDF would be a polynomial itself, we thought to elegantly prove its use with ZKP. We employed Bernstein polynomials [16] to approximate the PDF in a closed interval and, subsequently, performed rejection sampling. However, we encountered two problems: (i) our approximation was limited to a closed interval, whereas the Laplacian PDF has unbounded support, and (ii) the Bernstein coefficients are in general real numbers, which ZKP cannot process efficiently, and the propagation of errors when rounding with fixed precision is again complex to handle. Specifically, the complexity stems from the very high degree of the polynomials and the lack of homogeneity, i.e., there are many different degrees of monomials that all scale differently for a specific accuracy when multiplying inputs with a large power of 10 and rounding afterward.

Subsequently, we turned to the truncated geometric mechanism (TGM) [22], which coincidentally has the advantage to provide better accuracy for count queries [19], the focus of this study. Additionally, the truncation solved the problem of working with a closed interval (also a limitation of finite computers) by condensing the probability mass outside the interval in its lower and upper bounds. Moreover, the support of the geometric mechanism are integers, which ZKPs can process efficiently. Overall, TGM adapts to finite computers while still providing pure ϵ -DP. However, the probabilities assigned to each integer still fall on the real line. While we ensured these probabilities became rational numbers by carefully choosing ϵ [3], which a conventional computer can handle, the integers necessary to represent them were too large for the limited precision available in Circom [34] and other libraries for implementing ZKPs, and we were unable to write a theoretical bound for δ if we approximated the real numbers with finite precision.

To cope with precision limitations and the difficulty to bound δ , we then looked for simple sampling methods that provide bounds on δ , which finally led us to Dwork et al. [12] (see Section 3). This concluded our search, as their method for sampling exponentially distributed noise consists on repeatedly flipping unbiased coins (which is easily implemented in Circom with hashing and conversion to bit arrays), and provides a bound for δ based on the precision we can afford with Circom.

In our implementation, we used verifiable randomness co-created by the verifier (surveyor) and the prover (survey participant). As we noted in Section 2, when a surveyor repeatedly conducts the query in our implementation with different challenges, they could get additional information because by the law of large numbers, the truthful query value without noise can be determined with increasing accuracy. Consequently, in many scenarios, it may be appropriate to use a challenge that is hard coded, derived from the surveyor institution’s public key, or even derived solely from the attribute (e.g., the index of the age in the anonymous credential), such that repeated queries, even from different but colluding institutions, would not decrease the degree of plausible deniability ϵ . Furthermore, note that our verifiable randomized response implementation could be easily extended to flip biased coins by, e.g., generating a verifiable hash and checking whether its normalized value is lower than the desired bias.

7 CONCLUSION

We introduce primitives for implementing verifiable differentially private polls in the local setting. To achieve verifiability, we carefully selected DP mechanisms for binary and numerical data and adapted their implementations to SNARKs. Thanks to these primitives, we can achieve cryptographically verifiable survey responses while providing plausible deniability for survey participants and, in turn, not only reduce but entirely prevent bias in survey participants’ answers while giving them the needed privacy guarantees. Furthermore, note that our primitive for verifiable exponentially distributed noise allows for different aggregation queries beyond the count, as it can ingest arbitrary sensitivity – we limited our narrative to count queries for the simplicity of the explanations. Finally, thanks to the evaluations we performed, we conclude that practitioners can deploy our primitives with acceptable performance

We encourage practitioners to develop further primitives that can adapt to other DP mechanisms, e.g., the exponential mechanism for categorical data [42], and other randomized-response [6, 25, 30, 41, 58] and LDP [3, 31–33, 37, 38] approaches. Furthermore, conducting studies about how interviewees would perceive the built-in trust would allow the research community to understand how to frame polls and reassure candidates of their privacy. Lastly, improving the precision limitations of ZKP circuit compilers such as Circom and more literature on frameworks for bounding δ in *approximate* LDP would open the range of practical LDP mechanisms.

ACKNOWLEDGMENTS

We thank the Bavarian Ministry of Economic Affairs, Regional Development and Energy for their funding of the project “Fraunhofer Blockchain Center (20-3066-2-6-14)”, and the Ethereum foundation’s grant that made this paper possible.

REFERENCES

- [1] Fritz Alder, Jo Van Bulck, Jesse Spielman, David Oswald, and Frank Piessens. 2022. Faulty Point Unit: ABI Poisoning Attacks on Trusted Execution Environments. *Digital Threats: Research and Practice* 3, 2, Article 13 (2022), 26 pages. <https://doi.org/10.1145/3491264>
- [2] László Babai. 1985. Trading Group Theory for Randomness. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*. 421–429. <https://doi.org/10.1145/22145.22192>
- [3] Victor Balcer and Salil Vadhan. 2019. Differential Privacy on Finite Computers. *Journal of Privacy and Confidentiality* 9, 2 (2019). <https://doi.org/10.29012/jpc.679>
- [4] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. 2019. Scalable Zero Knowledge with No Trusted Setup. In *Annual International Cryptology Conference*. Springer, 701–732. https://doi.org/10.1007/978-3-030-26954-8_23
- [5] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. 2013. SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge. In *Annual Cryptology Conference*. Springer, 90–108. https://doi.org/10.1007/978-3-642-40084-1_6
- [6] Robert F. Boruch. 1971. Assuring Confidentiality of Responses in Social Research: A Note on Strategies. *The American Sociologist* 6, 4 (1971), 308–311. <http://www.jstor.org/stable/27701807>
- [7] Jan Camenisch and Anna Lysyanskaya. 2001. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 93–118. https://doi.org/10.1007/3-540-44987-6_7
- [8] Ran Canetti, Oded Goldreich, and Shai Halevi. 2004. The Random Oracle Methodology, Revisited. *J. ACM* 51, 4 (2004), 557–594. <https://doi.org/10.1145/1008731.10087340>
- [9] Jeffrey Champion, Abhi Shelat, and Jonathan Ullman. 2019. Securely Sampling Biased Coins with Applications to Differential Privacy. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. ACM, 603–614. <https://doi.org/10.1145/3319535.3354256>
- [10] Jing Chen and Silvio Micali. 2019. Algorand: A Secure and Efficient Distributed Ledger. *Theoretical Computer Science* 777 (2019), 155–183.

- [11] Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, and Bryan Parno. 2016. Cinderella: Turning Shabby X.509 Certificates into Elegant Anonymous Credentials with the Magic of Verifiable Computation. In *Symposium on Security and Privacy*. IEEE, 235–254. <https://doi.org/10.1109/SP.2016.22>
- [12] Cynthia Dwork, Krishnamurthy Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. 2006. Our Data, Ourselves: Privacy Via Distributed Noise Generation. In *Advances in Cryptology* (2006), Serge Vaudenay (Ed.), Vol. 4004. Springer, 486–503. https://doi.org/10.1007/11761679_29
- [13] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating Noise to Sensitivity in Private Data Analysis. In *Theory of Cryptography*, Shai Halevi and Tal Rabin (Eds.). Springer, 265–284. https://doi.org/10.1007/11681878_14
- [14] Cynthia Dwork and Aaron Roth. 2013. The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science* 9, 3-4 (2013), 211–407. <https://doi.org/10.1561/04000000042>
- [15] European Commission. 2021. Commission Proposes a Trusted and Secure Digital Identity for all Europeans. https://ec.europa.eu/commission/presscorner/detail/en/ip_21_2663
- [16] Rida T. Farouki. 2012. The Bernstein Polynomial Basis: A Centennial Retrospective. 29, 6 (2012), 379–419. <https://doi.org/10.1016/j.cagd.2012.03.001>
- [17] Amos Fiat and Adi Shamir. 1986. How to Prove Yourself: Practical Solutions to Identification and Signature Problems. In *Conference on the Theory and Application of Cryptographic Techniques*. Springer, 186–194. https://doi.org/10.1007/3-540-47721-7_12
- [18] Georg Fuchsbaue. 2018. Subversion-Zero-Knowledge SNARKs. In *IACR International Workshop on Public Key Cryptography*. Springer, 315–347.
- [19] Gonzalo Munilla Garrido, Joseph Near, Aitsam Muhammad, Warren He, Roman Matzutt, and Florian Matthes. 2021. Do I Get the Privacy I Need? Benchmarking Utility in Differential Privacy Libraries. <http://arxiv.org/abs/2109.10789>
- [20] Gonzalo Munilla Garrido, Johannes Sedlmeir, Omer Uludağ, Ilias Soto Alaoui, Andre Luckow, and Florian Matthes. 2021. Revealing the Landscape of Privacy-Enhancing Technologies in the Context of Data Markets for the IoT: A Systematic Literature Review. <https://arxiv.org/abs/2107.11905>
- [21] Ivan Gazeau, Dale Miller, and Catuscia Palamidessi. 2016. Preserving Differential Privacy under Finite-precision Semantics. *Theoretical Computer Science* 655 (2016), 92–108. <https://doi.org/10.1016/j.tcs.2016.01.0150>
- [22] Arpita Ghosh, Tim Roughgarden, and Mukund Sundararajan. 2012. Universally Utility-maximizing Privacy Mechanisms. *SIAM J. Comput.* 41, 6 (2012), 1673–1693. <https://doi.org/10.1137/09076828X>
- [23] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. 1989. The Knowledge Complexity of Interactive Proof Systems. *SIAM J. Comput.* 18, 1 (1989), 186–208. <https://doi.org/10.1137/0218012>
- [24] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. 2021. Poseidon: A New Hash Function for Zero-Knowledge Proof Systems. In *30th USENIX Security Symposium*. 519–535. <https://www.usenix.org/system/files/sec21-grassi.pdf>
- [25] Bernard G. Greenberg, Abdel-Latif A. Abul-Ela, Walt R. Simmons, and Daniel G. Horvitz. 1969. The Unrelated Question Randomized Response Model: Theoretical Framework. *J. Amer. Statist. Assoc.* 64, 326 (1969), 520–539. <http://www.jstor.org/stable/2283636>
- [26] Jens Groth. 2016. On the Size of Pairing-based Non-interactive Arguments. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 305–326. https://doi.org/10.1007/978-3-662-49896-5_11
- [27] Jens Groth, Rafail Ostrovsky, and Amit Sahai. 2006. Perfect Non-Interactive Zero Knowledge for NP. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 339–358. https://doi.org/10.1007/11761679_21
- [28] Hermez Network. 2021. Open Sourcing an Ultra-fast zk Prover: Rapidsnark. <https://blog.hermez.io/open-sourcing-ultra-fast-zk-prover-rapidsnark/>
- [29] Naohise Holohan and Stefano Braghin. 2021. Secure Random Sampling in Differential Privacy. In *Computer Security*, Elisa Bertino, Haya Shulman, and Michael Waidner (Eds.). Vol. 12973. Springer, 523–542. https://doi.org/10.1007/978-3-030-88428-4_26
- [30] Naohise Holohan, Douglas J. Leith, and Oliver Mason. 2015. Differential privacy in metric spaces: Numerical, Categorical and Functional Data under the One Roof. *Information Sciences* 305 (2015), 256–268. <https://doi.org/10.1016/j.ins.2015.01.021>
- [31] Naohise Holohan, Douglas J. Leith, and Oliver Mason. 2017. Extreme Points of the Local Differential Privacy Polytope. *Linear Algebra Appl.* 534 (2017), 78–96. <https://doi.org/10.1016/j.laa.2017.08.011>
- [32] Naohise Holohan, Douglas J. Leith, and Oliver Mason. 2017. Optimal Differentially Private Mechanisms for Randomised Response. *IEEE Transactions on Information Forensics and Security* 12, 11 (2017), 2726–2735. <https://doi.org/10.1109/TIFS.2017.2718487>
- [33] Justin Hsu, Sanjeev Khanna, and Aaron Roth. 2012. Distributed Private Heavy Hitters. In *Proceedings of the 39th International Colloquium Conference on Automata, Languages, and Programming – Volume Part I*. Springer, 461–472. https://doi.org/10.1007/978-3-642-31594-7_39
- [34] iden3. 2018. Circom. <https://docs.circom.io/>
- [35] Intel. 2022. 12th Generation Intel Core Processors. <https://cdrdv2.intel.com/v1/dl/getContent/655258>
- [36] Lefki Kacem and Catuscia Palamidessi. 2018. Geometric Noise for Locally Private Counting Queries. In *Proceedings of the 13th Workshop on Programming Languages and Analysis for Security*. ACM, 13–16. <https://doi.org/10.1145/3264820.3264827>
- [37] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. 2014. Extremal Mechanisms for Local Differential Privacy. In *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger (Eds.), Vol. 27. <https://proceedings.neurips.cc/paper/2014/file/86df7dcfd896fcdf2674f757a2463eba-Paper.pdf>
- [38] Vishesh Karwa, Aleksandra B Slavković, and Pavel Krivitsky. 2014. Differentially Private Exponential Random Graphs. In *International Conference on Privacy in Statistical Databases*. Springer, 143–155. https://doi.org/10.1007/978-3-319-11257-2_12
- [39] Fumiyuki Kato, Yang Cao, and Masatoshi Yoshikawa. 2021. Preventing Manipulation Attack in Local Differential Privacy Using Verifiable Randomization Mechanism. <https://arxiv.org/abs/2104.06569>
- [40] Fatima Khalid and Ammar Masood. 2022. Vulnerability Analysis of Qualcomm Secure Execution Environment. *Computers & Security* 116 (2022), 102628. <https://doi.org/10.1016/j.cose.2022.102628>
- [41] N. S. Mangat and Ravindra Singh. 1990. An Alternative Randomized Response Procedure. *Biometrika* 77, 2 (1990), 439–442. <http://www.jstor.org/stable/2336829>
- [42] Frank McSherry and Kunal Talwar. 2007. Mechanism Design via Differential Privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, 94–103. <https://doi.org/10.1109/FOCS.2007.66>
- [43] Ilya Mironov. 2012. On Significance of the Least Significant Bits for Differential Privacy. In *Proceedings of the Conference on Computer and Communications Security*. ACM. <https://doi.org/10.1145/2382196.2382264>
- [44] Arjun Narayan, Ariel Feldman, Antonis Papadimitriou, and Andreas Haeberlen. 2015. Verifiable Differential Privacy. In *Proceedings of the 10th European Conference on Computer Systems*. Article 28. <https://doi.org/10.1145/2741948.2741978>
- [45] OMTP. 2009. Advanced Trusted Environment: OMTP TR1. , 204 pages. <http://www.gsma.com/newsroom/wp-content/uploads/2012/03/omtpadvancedtrustedenvironmentomtptr1v11.pdf>
- [46] Alexander Rieger, Tamara Roth, Johannes Sedlmeir, Linda Weigl, and Gilbert Fridgen. 2022. Not Yet Another Digital Identity. *Nature Human Behaviour* 6, 1 (2022), 3–3. <https://doi.org/10.1038/s41562-021-01243-0>
- [47] Timon Rückel, Johannes Sedlmeir, and Peter Hofmann. 2022. Fairness, Integrity, and Privacy in a Scalable Blockchain-Based Federated Learning System. *Computer Networks* 202 (2022), 108621.
- [48] Sebastian Sartor, Johannes Sedlmeir, Alexander Rieger, and Tamara Roth. 2022. Love at First Sight? A User Experience Study of Self-Sovereign Identity Wallets. In *Proceedings of the 30th European Conference on Information Systems*. AIS.
- [49] Martin Schanzbach, Thomas Kilian, Julian Schütte, and Christian Banse. 2019. ZKclaims: Privacy-preserving Attribute-Based Credentials using Non-Interactive Zero-Knowledge Techniques. <https://arxiv.org/abs/1907.09579>
- [50] Vincent Schlatt, Johannes Sedlmeir, Simon Feulner, and Nils Urbach. 2021. Designing a Framework for Digital KYC Processes Built on Blockchain-Based Self-Sovereign Identity. *Information & Management* (2021), 103553. <https://doi.org/10.1016/j.im.2021.103553>
- [51] Johannes Sedlmeir, Jonathan Lautenschlager, Gilbert Fridgen, and Nils Urbach. 2022. The Transparency Challenge of Blockchain in Organizations. *Electronic Markets* (2022). <https://doi.org/10.1007/s12525-022-00536-0>
- [52] Johannes Sedlmeir, Reilly Smethurst, Alexander Rieger, and Gilbert Fridgen. 2021. Digital Identities and Verifiable Credentials. *Business & Information Systems Engineering* 63, 5 (2021), 603–613.
- [53] Gerardo I Simari. 2002. A Primer on Zero Knowledge Protocols. (2002), 12. <http://cs.uns.edu.ar/~gis/publications/zkp-simari2002.pdf>
- [54] Dimitrios Skarlatos, Mengjia Yan, Bhargava Gopireddy, Read Sprabery, Josep Torrellas, and Christopher W. Fletcher. 2019. MicroScope: Enabling Microarchitectural Replay Attacks. In *Proceedings of the 46th International Symposium on Computer Architecture*. ACM, 318–331. <https://doi.org/10.1145/3307650.3322228>
- [55] Georgia Tsaloli and Aikaterini Mitrokotsa. 2019. Differential Privacy Meets Verifiable Computation: Achieving Strong Privacy and Integrity Guarantees. In *6th International Joint Conference on e-Business and Telecommunications*. 425–430. <https://doi.org/10.5220/0007919404250430>
- [56] Lun Wang, Ruoxi Jia, and Dawn Song. 2022. D2P-Fed: Differentially Private Federated Learning With Efficient Communication. (2022). <https://arxiv.org/abs/2006.13039>
- [57] Stanley L Warner. 1965. Randomized Response: A Survey Technique for Eliminating Evasive Answer Bias. *J. Amer. Statist. Assoc.* 60, 309 (1965), 63–69. <https://doi.org/10.1080/01621459.1965.10480775>
- [58] Qing Yang, Wei Yu, and Yacine Challal (Eds.). 2016. *Wireless Algorithms, Systems, and Applications*. Lecture Notes in Computer Science, Vol. 9798. Springer. <https://doi.org/10.1007/978-3-319-42836-9>