PINE: Efficient Verification of a Euclidean Norm Bound of a Secret-Shared Vector

Guy N. Rothblum, *Apple;* Eran Omri, *Ariel University and Ariel Cyber Innovation Center;* Junye Chen and Kunal Talwar, *Apple*

https://www.usenix.org/conference/usenixsecurity24/presentation/rothblum

This paper is included in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

Open access to the Proceedings of the 33rd USENIX Security Symposium is sponsored by USENIX.

PINE: Efficient Verification of a Euclidean Norm Bound of a Secret-Shared Vector*

Guy N. Rothblum
Apple

Eran Omri†
Ariel University
Ariel Cyber Innovation Center

Junye Chen
Apple

Kunal Talwar
Apple

Abstract

Secure aggregation of high-dimensional vectors is a fundamental primitive in federated statistics and learning. A two-server system such as PRIO allows for scalable aggregation of secret-shared vectors. Adversarial clients might try to manipulate the aggregate, so it is important to ensure that each (secret-shared) contribution is well-formed. In this work, we focus on the important and well-studied goal of ensuring that each contribution vector has bounded Euclidean norm. Existing protocols for ensuring bounded-norm contributions either incur a large communication overhead, or only allow for approximate verification of the norm bound. We propose **P**rivate **I**nexpensive **N**orm **E**nforcement (PINE): a new protocol that allows exact norm verification with little communication overhead. For high-dimensional vectors, our approach has a communication overhead of a few percent, compared to the 16-32x overhead of previous approaches.

1 Introduction

Data analyses and machine learning on user data have the potential to improve various applications, but might raise concerns about users' privacy. Fortunately, these analyses can be performed in a federated setting [Kai+21; LSTS20] while ensuring strong formal privacy guarantees. This has sparked significant interest in developing techniques and infrastructures to support private federated data analyses. In this work, we focus on a fundamental primitive: estimating the mean (or equivalently, the sum) of a set of high-dimensional vectors. This primitive is natural and important in its own right, and is also a basic building block for several tasks (e.g. [RNFH19; RZHP20]). Most notably, it is fundamental for federated optimization of machine learning models. Furthermore, several other ML tasks such as PCA, k-means and EM can be reduced to (multiple applications of) aggregation.

Secure Aggregation techniques enable strong privacy guarantees without requiring trust in a single server. Various approaches have been studied for designing such systems. One approach is to have client devices run a secure multiparty computation to compute the aggregate, which has been explored in [DKMMN06; BIKMMPRSS17; SGA20; SGA21; BBGLR20]. Another approach relies on two or more servers to perform the aggregation, using protocols that ensure that no single server can learn anything beyond the final aggregate (security holds so long as the two servers do not collaborate). A common approach is for each client to secret-share its contribution between the two servers, and for the servers to use a secure protocol that allows them to learn the aggregate, but nothing more about any individual client's contribution. We refer to this approach as *distributed aggregation*. The PRIO system [CB17] proposed a protocol for computing arbitrary functions over secret-shared data in the presence of two or more servers. This approach has been further developed in subsequent research [BBCGI19; BBCGI21; BGGKMRS22; CP22; BBCGI23; Tal+23], applied in practice [HMR18; AG21] and forms the basis of an IETF [GPRW23] draft standard for aggregation. In this work, we focus on distributed aggregation of high-dimensional integer[1] vectors.

Real-world implementations of secure aggregation must deal with adversarial clients attempting to manipulate the results. An important advantage of the PRIO approach is that it allows the servers to validate certain properties of the client contributions, while preserving a zero-knowledge (ZK) property, meaning that the servers learn nothing beyond the aggregate and the fact that the client shares were valid. A rich line of research has focused on developing efficient methods to verify various properties of client contributions. In the context of summing up vectors in $\mathbb{R}^d$, a natural objective is verifying that each client contribution has bounded Euclidean norm. The Euclidean norm constraint arises naturally in many machine

*Extended abstract. See [ROCT23] for the full version.

†Part of this work was performed while EO was at Apple.

[1] While one is often interested in vectors of floating point values, standard techniques can translate the problem to fixed-precision vectors. Scaling then turns those into integer vectors from a bounded range. See the discussion in Sections 1.1 and 5.

learning settings [Dos+21; Deh+23; ZLLBRZGS23]. Many model poisoning attacks (e.g. [BBG19; FCJG20; SH21]) rely on clients submitting vectors of large norm (so called "boosted gradients"). Recent works [SHKR22; SKSM19] have shown that ensuring a bounded ℓ_2 norm is effective against a large family of realistic poisoning attacks in the federated learning setting.[2]

In this work, we focus on zero-knowledge protocols for verifying the bounded Euclidean norm property. Such protocols require the client and the servers to perform additional computations and communication, where different works obtain different overheads. One important measure is the communication cost, especially in settings where the vectors involved can be of high dimensions (e.g. [RMRB19; Pau+21; Xu+23; XZACKMRZ23; PAFSTL23]). We review the works most related to our contributions.

One approach taken by prior work [CB17; BBCGI19] and in proposed implementations in the IETF standard [ISR23; RU23] has the client secret-share each bit of each coordinate of its vector as an element in a finite field. For d-dimensional vectors with b bits of precision, this requires the client to send at least *bd field elements* to each server. The field size itself must be rather large to keep the soundness error small. In high-dimensional settings, this translates to a substantial communication overhead. For example, in the implementation in [RU23], the communication overhead is at least 16x (compared to secret sharing without validity proofs). We note that the servers also need to verify that the secret shares are indeed of bits, but the communication needed for this latter task can be smaller [BBCGI19].

In PRIO+ [AGJOP22], the communication from the client to each server is only *bd bits* (rather than field elements), but this requires an offline setup phase that involves expensive cryptographic operations (it also gives a weaker norm bound guarantee). In ELSA [RSWP22], the setup avoids expensive cryptographic operations, but requires more communication from the clients. We elaborate on the comparison with these works below, but the main distinction is that our work focuses on a setting where there is no expensive setup, and we aim to minimize the communication from the clients to the servers.

With the goal of minimizing the client-to-server communication, the work of [Tal22] relaxed the zero-knowledge guarantee to differential ZK and showed a more communication-efficient protocol. However, that protocol only ensured approximate norm verification, limiting its usefulness in some settings. The protocol allows for a trade-off between the soundness and completeness errors and the approximation guarantee, but even for relatively permissive error thresholds (say, 0.01 soundness and completeness errors), the protocol might accept vectors of norm 50x the target bound. This reduces its effectiveness in dealing with malicious clients.

[2]No method can completely prevent model poisoning [SHKR22] (for a theoretical perspective, see e.g. [MM17]); bounding the ℓ_2 norm limits the impact of adversarial clients.

1.1 Our Work

Our main contribution is PINE (for **P**rivate **I**nexpensive **N**orm **E**nforcement): a communication-efficient protocol for Euclidean norm verification that can verify the exact norm bound with a strong (statistical) zero knowledge guarantee and with no offline setup. Zero knowledge holds even against malicious behavior by a server (we always assume the servers do not collaborate, i.e. at least one of them is honest). Theoretically, our verification protocol requires the client to communicate only $\tilde{O}(\sqrt{d})$ additional field elements. In practice, for typical parameter settings, this overhead is a small fraction of the communication needed to secret share d field elements.

Theorem 1.1 (Informal version of Theorem 3.15)**.** *Let $X \in \mathbb{Z}^d$ be a secret-shared vector and $B \geq 0$. Fix $\rho > 0$, and set $r = \lceil 32 \ln \frac{1}{\rho} \rceil$. For any field of size $q \geq \Omega(\max\{B, 3r\})$ there is a distributed verification protocol with the following properties:*

1. ***Completeness:*** *If $\sum_i X_i^2 \leq B$, the verifiers accept with probability at least $1 - \rho$.*
2. ***Soundness:*** *If $\sum X_i^2 > B$ (over the integers), the probability that the verifiers accept is at most $\rho + O(\sqrt{d}/q)$.*
3. ***Zero-Knowledge****: The protocol satisfies distributed statistical zero-knowledge: the view of each verifier can be simulated up to statistical distance ρ. This guarantee holds even for a single malicious verifier (so long as the other verifier follows the protocol).*

The proof system is in the common reference string model, and consists of a single message of length $O\left(\sqrt{d}\log q + r\log^2 q\right)$ sent by the prover to each verifier.

The protocol is derived from the interactive 4-message protocol of Theorem 3.15 by using a variant of the Fiat-Shamir heuristic suited for the distributed verification setting (see the full version [ROCT23]).[3] While we state our results for the case of two verifiers, our approach is modular and can be used with multiple verifiers. Concrete performance evaluation is below and in Section 5.

Statistical PINE: Overview. A central difficulty in efficient verification of Euclidean norms is that PRIO relies on arithmetic over a finite field, whereas our property of interest deals with arithmetic over integers/reals. One can use known techniques [BBCGI19] to efficiently verify that the sum of of squared entries *modulo the field size* satisfies a certain bound. If the coordinates are all small enough, then the squared norm modulo the field sizes equals the squared norm over the integers, so the preceding check suffices. However, it is challenging to validate smallness of coordinates in their natural representation as field elements. Existing approaches achieve

[3]The soundness error is proved for the interactive protocol, and should be set to be sufficiently small to account for the Fiat-Shamir transform.

this by encoding the coordinates in their binary representation, which can enforce the requisite smallness.

We take a different approach. Instead of encoding the coordinates of a vector in their binary representation, we devise a randomized test that can detect whether there is a "wraparound" when computing the sum of squares over a finite field. We first verify that the sum of squared entries modulo the field size q is in the range $[0,B]$. For large enough q, this gives us a promise problem: either the sum of squared entries (over integers) is at most B, or is at least q. We test for this *wraparound* by taking a random dot product with a $-1,0,1$ vector, and we demonstrate that if there is wraparound, the dot product is likely to be large. On the other hand, if the sum of the squared entries was small to begin with, then the dot product will be small with high probability. These bounds on the dot product, which are our main technical contribution, are proved using a delicate case analysis on the vector's infinity norm. The challenging case (small infinity norm) is analyzed using the Berry-Esseen theorem.

Thus, we reduce the problem of proving a bound on the squared norm to proving boundedness of a few scalars (the outcomes of independent dot products), which can be done at a small overhead. Our communication overhead is dominated by the communication needed to verify the sum of squared entries over the finite field; the $\sqrt{d}$ term can be further reduced to $d^{1/c}$ by using an $c+O(1)$-round protocol (the additional interaction can be eliminated using the Fiat-Shamir transform).

Differential ZK. We also show a simpler scheme that relaxes ZK to Differential ZK: intuitively, the secrecy of the client's contribution is protected via a *differential privacy* [DMNS06] guarantee (rather than the perfect or statistical guarantees that are more common in the literature). Beyond its simplicity, the protocol also achieves smaller communication in some parameter regimes (especially when the number of dimensions is not too large). We note that the relaxation to differential ZK can be quite reasonable, since differential privacy is often all that is guaranteed given that the (approximate) results of the entire aggregation are to be made public to the servers.

Theorem 1.2 (Informal Version of Theorem 4.3)**.** *Let $X \in \mathbb{Z}^d$ be a secret-shared vector. Let $\varepsilon,\delta \in (0,1)$ and $B \geq 1$. For a field of size $q > 4\left(\sqrt{B}+\sqrt{d}+\sqrt{dB\frac{2\ln 2.5/\delta}{\varepsilon}\cdot(1+\frac{2\sqrt{\log 8e/\delta}}{\sqrt{d}}+\frac{2\log 8e/\delta}{d})}\right)^2$, there is a distributed verification protocol with:*

1. ***Completeness:** If $\sum_{i=1}^d X_i^2 \leq B$, then the verifiers accept with probability* 1.
2. ***Soundness:** If $\sum X_i^2 > B$ (over the integers), the probability that the verifiers accept is at most $O(\sqrt{d}/q)$.*
3. ***Zero-Knowledge**: The protocol satisfies (ε,δ)-differential zero knowledge: the view of each verifier can be efficiently simulated up to (ε,δ)-closeness. This guarantee holds even for a single malicious verifier (so long as the other verifier follows the protocol).*

In addition to the secret shares of x, the client sends a proof of length $(O(\sqrt{d})+O(\log_2 q))\cdot\lceil\log_2 q\rceil$.

This protocol is derived from an interactive 3-message protocol a distributed-verification variant of the Fiat-Shamir heuristic, see the full version [ROCT23].

Performance analysis. We analyze the performance of our protocols in terms of the communication overhead, beyond the communication that is needed to simply send secret shares for distributed aggregation (without any robustness to poisoning attacks). In Section 5, we provide analyses for several choices of parameters. Here, in Table 1, we highlight the performance for a typical choice of parameters, where we aggregate d-dimensional integer vectors of ℓ_2 norm at most 2^{15} with $d \in \{10^4,10^5,10^6,10^7\}$. We work over a field of size $\approx 2^{64}$ and use soundness and zero-knowledge error 2^{-50}.

As seen in Table 1, our statistical zero-knowledge protocol achieves small overhead even when the number of dimensions is as small as 10^4, and the communication overhead becomes negligible as the number of dimensions grows. Comparing with prior work [BBCGI19; ISR23], the overhead is reduced by a multiplicative factor of between 70x (when the number of dimensions is only 10,000) to 10^4x (when the number of dimensions is as high as 10^7).

Our differential ZK protocol achieves even smaller overheads so long as the number of dimensions is not huge (albeit, the secrecy guarantee is more relaxed). Once the number of dimensions grows to 10^7, PINE with Differential ZK requires a larger field size (9 bytes instead of 8 bytes), this incurs an initial overhead of $\frac{1}{8}$ for secret-sharing the data over a larger field. Previous work with differential ZK [Tal22], would have similar increased field size requirements for these parameters. While their protocol would not have any additional overhead, it only gives a weaker robustness guarantee that vectors that are 50x the norm bound are rejected with probability 0.99.

We briefly elaborate on our choice of the norm bound: a typical setting when aggregating gradients is that we aggregate floating point vectors of Euclidean norm at most 1. We can convert these floating point vectors to integer vectors by multiplying by 2^b and rounding. With $b=15$ bits of precision per co-ordinate, this translates the problem to verifying ℓ_2 norm bound $\sqrt{B}=2^{15}$. This is typically sufficient for high dimensional vectors. Working over a field of size about $\approx 2^{64}$ suffices to allow aggregating millions of such vectors (whereas field size $\approx 2^{32}$ would not be sufficient for 100K vectors). We refer the reader to Section 5 for further elaboration and for performance evaluation in other parameter regimes.

Further comparison to PRIO+ and ELSA. In Table 2, we provide a qualitative comparison to the most closely related

	$d = 10^4$	$d = 10^5$	$d = 10^6$	$d = 10^7$
no robustness, # bits sent	$64 \cdot 10^4$	$64 \cdot 10^5$	$64 \cdot 10^6$	$64 \cdot 10^7$
prior work, overhead [BBCGI19; ISR23]	$> 1500\%$	$> 1500\%$	$> 1500\%$	$> 1500\%$
PINE, Statistical ZK, overhead	22%	3.18%	0.49%	0.13%
PINE, Differential ZK, overhead	4.77%	1.46%	0.32%	12.63%

Table 1: Communication analysis: our protocols and prior work. Parameters: field size $q \approx 2^{64}$ for aggregation, d-dimensional data, soundness error 2^{-50}, zero-knowledge error $\delta = 2^{-50}$. For differential ZK $\varepsilon = 0.1$.

Protocol	Linear online	Linear offline	Expensive crypto
Prio3 [BBCGI19; ISR23]	N	Y	N
Prio+ [AGJOP22]	Y	N	Y
ELSA [RSWP22]	Y	N	N
PINE (our work)	Y	Y	N

Table 2: Qualitative comparison to representative prior works. All works are in the distributed two-server trust model. *Linear offline and online* mean that the communication is dominated by the cost of communicating at most d field elements for the field where aggregation should occur, either in an offline stage (before the client's contribution is specified), or in the online stage (respectively). *Expensive crypto* means that the protocol uses operations such as oblivious transfer.

works to ours. In particular, we consider works in the distributed aggregation setting, with two non-colluding servers and privacy against (one out of two) malicious servers. The PRIO+ system [AGJOP22] reduces the communication cost of sharing the client's vector to bd bits (where 2^b bounds the magnitude of each entry). This is significantly smaller than our protocol, which requires sending d field elements (e.g. in Table 1 each field element is 64 bits, whereas $b = 15$). However, PRIO+ requires an expensive offline setup between the servers, which perform cryptographic oblivious transfer operations and exchange more communication than d field elements per client. Moreover, verifying a bound on the Euclidean norm would also require a subsequent online cryptographic protocol (and there is further overhead for malicious-server security, see [RSWP22]). Other works, such as [HLXCZ21; HKJ20], also make use of more advanced cryptographic operations. PINE avoids an expensive setup and cryptographic operations of this type. The ELSA system [RSWP22] also avoids the use of expensive cryptographic operations, but replaces them with communication from the client, which sends more than d field elements to the servers. The expensive communication from the client to the servers can be performed offline, but it is a large communication compared to our work. Further, in many PRIO-like settings, anonymous clients engage in a one-shot interaction with the servers, so an offline setup is not appropriate.

Further related work. In a very recent work Boneh *et al.* [BBCGI23] show how *arithmetic sketching schemes* can be used to design efficient protocols for verifying properties of secret-shared data. They show, however, that the linear sketches at the heart of their technique cannot be used to verify L_2-norm constraints (or any L_p norm for $p > 1$).

2 Model, Definitions and Preliminaries

Definition 2.1. *The statistical distance between two finite random variables X and Y is*

$$\mathrm{SD}(X,Y) = \frac{1}{2}\sum_a |\Pr[X = a] - \Pr[Y = a]|\,.$$

Two ensembles $X = \{X_n\}_{n\in\mathbb{N}}$ and $Y = \{Y_n\}_{n\in\mathbb{N}}$ are said to be statistically close, denoted $X \overset{s}{\equiv} Y$, if there exists a negligible function $\mu(\cdot)$, such that for all $n \in \mathbb{N}$, it holds that $\mathrm{SD}(X_n, Y_n) \le \mu(n)$.

Runtimes and field operations over $\mathbb{GF}[q]$. We measure the prover's and the verifiers' runtimes by the number of field operations we perform: we usually count addition and multiplication as a single field operation, and also allow other basic atomic operations such as translating field elements to their natural representation as bit vectors and vice versa. In performance evaluations, we also measure the number of multiplications as a primary complexity measure (as done in prior work, since these are significantly more expensive than other field operations).

Secret sharing over $\mathbb{GF}[q]$. In the distributed verification setting we study, a client (or prover) secret-shares data between two or more servers (or verifiers). Each secret-shared value is an element $\alpha \in \mathbb{GF}[q]$ from a field, and the secret shares are random field elements whose sum is α ("arithmetic shares"). In particular, each server's share is, on its own, a uniformly random field element.

2.1 Distributed Verification Protocols

As discussed above, we study a distributed model, in which a single prover (or client) $\mathbf{P}$ interacts with two verifiers (or servers) $\mathbf{V}_0, \mathbf{V}_1$ over a complete network with secure point-to-point channels. We design several protocols in this setting,

and then compose these protocols to obtain the PINE proof systems. Our distributed verification protocols have inputs and outputs of the following type:

Common (Public) Inputs: All three parties hold the same common and publicly known parameter vector $\bar{\mathbf{T}}$. Such parameters may include, field size, scalars, protocol parameters.

Private Inputs: The prover holds some vector of input values $\bar{\mathbf{I}}$. Each such value I is secret-shared by the two verifiers such that $\mathbf{V}_j$ holds the share $[I^{(j)}]$.

Common (Public) Output: The common output of the two verifiers includes a single bit, indicating whether they accepted or rejected the proof. The verifiers can also output additional common outputs.

Private Outputs: At the end of the interaction, the prover outputs new private values. Each such value $\bar{\mathbf{O}}$ is secret-shared by the two verifiers, and these are called the shared outputs of the verifiers.

We define distributed interactive zero-knowledge proof protocols (dZKIPs) for this setting. These protocols guarantee completeness and soundness based on a condition on the common and private inputs (this "input condition" is formalized by requiring that the combined input is in a (pair) language). The zero knowledge property is also guaranteed for inputs that satisfy this input condition (are in the language). Some of the protocols also have an "output condition" on the common and private outputs (also formalized as membership in an output pair language, this condition can be empty). Completeness means that if the inputs satisfy the input condition and all parties follow the protocol, then w.h.p. the verifiers should accept and the common and private outputs (if any) should be in the output language. Soundness means that if the input condition is violated, then w.h.p. either the verifiers reject (in their common output), or the output condition is violated. Note that the verifiers might not "know" that the output condition is violated (since they only see secret shares of the private outputs): in our work this will be detected by a subsequent protocol. Allowing for private inputs and outputs, and for general input and output conditions is helpful for designing modular sub-protocols that can later be composed.

Our protocols also guarantee a strong zero-knowledge property: so long as the input is in the input language, and the verifiers are honest, both verifiers learn nothing from executing the protocol. Formally, each verifier's view in a protocol execution with the honest prover and a second verifier who follows the protocol can be simulated efficiently (the view includes the common inputs, its secret shares of the private inputs, random coins, messages received, and its secret shares of private outputs). We remark that in the classical setting for (non-distributed) interactive zero-knowledge proofs, a single prover $\mathbf{P}$ interacts with a single verifier $\mathbf{V}$ over a common input X. In such an interaction, the aim of the prover is to convince the verifier that $X \in L$ for some language L, where the verifier knows what X is, and zero-knowledge means that the verifier should not learn anything beyond X and the fact of its membership in L. In contrast, in our (distributed) setting, the statement X is not known to any single verifier (since it is secret-shared), and the zero-knowledge requirement means that any single verifier should learn nothing beyond X's membership in L.

Definition 2.2 (Distributed ZK Interactive Proof (dZKIP))**.** *We say that a 2-verifier interactive proof protocol* $\Pi = (\mathbf{P}; \mathbf{V}_0; \mathbf{V}_1)$ *is a distributed (strong) zero-knowledge proof for an input (pair) language* L_{inp} *and an output language* L_{out} *if* Π *satisfies the following:*

- α***-Completeness.*** *If the common and private inputs are in the input language, i.e.* $(\bar{\mathbf{T}}, \bar{\mathbf{I}}) \in \mathbf{L}_{\mathsf{inp}}$ *and the prover and the verifiers follow the protocol, then with all but* α *probability the verifiers accept and the private outputs satisfy the output condition, i.e.* $\bar{\mathbf{O}} \in \mathbf{L}_{\mathsf{out}}$ *(the probability is over all coins tossed by all parties in the protocol).*

- β***-Soundness.*** *If the common and private inputs are* not *in the language, i.e.* $(\bar{\mathbf{T}}, \bar{\mathbf{I}}) \notin \mathbf{L}_{\mathsf{inp}}$*, then for any adversarial cheating prover strategy, with all but* β *probability, either the verifiers reject, or the private outputs violate the output condition, i.e.* $\bar{\mathbf{O}} \notin \mathbf{L}_{\mathsf{out}}$ *(the probability is over the verifiers' coin tosses. The cheating prover is deterministic w.l.o.g).*

- γ***-Strong Distributed Honest-Verifier Zero-Knowledge (dZK)*** *(see [BBCGI19]). There exists an efficient simulator* S*, such that for every input pair* $(\bar{\mathbf{T}}, \bar{\mathbf{I}}) \in \mathbf{L}_{\mathsf{inp}}$*, for every* $j \in \{0,1\}$ *and every 2-out-of-2 sharing* $[\bar{\mathbf{I}}] = \left(\bar{\mathbf{I}}^{(\mathbf{0})}, \bar{\mathbf{I}}^{(\mathbf{1})}\right)$*, the view of* $\mathbf{V}_j$ *in an execution with the honest prover and* $\mathbf{V}_{1-j}$ *is* γ*-statistically close to the output of the simulator* S *on input* $\left(j, \bar{\mathbf{T}}, \bar{\mathbf{I}}^{(\mathbf{j})}\right)$*.*

By default we take $\alpha = 0$ *(perfect completeness) unless we explicitly note otherwise. Similarly, by default* $\gamma = 0$ *(perfect distributed zero-knowledge). We say the protocol is* public-coins *if all the messages sent from the verifiers to the prover are random coin tosses. We also require that in a public-coins protocol, the communication between the two verifiers consists solely of a single simultaneous message exchange: after the interaction with the prover is complete,* $\mathbf{V}_0$ *sends a single message to* $\mathbf{V}_1$*, and (at the same time)* $\mathbf{V}_1$ *sends a single message to* $\mathbf{V}_0$*. These messages should not depend on each other (this property is important for zero-knowledge of the Fiat-Shamir transform, see [ROCT23]).*

We sometimes consider dZKIPs for *promise problems*, where the completeness condition and the soundness condition apply to disjoint sets (rather than to the language L_{inp}

and its complement). See, for example, the "wraparound protocol" of Section 3.2.

Remark 2.3 (indistinguishability under varying private inputs)**.** *The dZK property implies that, for a fixed public input $\bar{\mathbf{T}}$, we can consider different private inputs that are in the pair language, and each verifier's views will be β-statistically close under these varying inputs, so long as its share remains unchanged. For example, fixing $\mathbf{V}_0$'s share to be $\bar{\mathbf{I}}^{(\mathbf{0})}$, we can consider an execution where V_1's share is $\bar{\mathbf{I}}^{(\mathbf{1})}$ and another execution where V_1's share is $\bar{\mathbf{I}}'^{(1)}$, and $\mathbf{V}_0$'s views in these executions will be statistically close, so long as the underlying private inputs $\bar{\mathbf{I}},\bar{\mathbf{I}}'$ are both in the pair language (w.r.t. the fixed public input $\bar{\mathbf{T}}$).*

Remark 2.4 (Non-interactive and malicious ZK via Fiat-Shamir)**.** *We construct and analyze our protocols as interactive proof systems with honest-verifier zero-knowledge guarantees, as formalized in Definition 2.2. These protocols can be transformed to non-interactive protocols that guarantee zero-knowledge against (one out of two) malicious verifiers using the Fiat-Shamir transform for the distributed setting.*

2.2 Composition of dZKIPs

Distributed ZKIP protocols maintain their zero-knowledge properties under (sequential) composition, so long as the protocols have the property that for every common input, for every input share there exists a completion of the share to an input on which the verifiers accept.

Lemma 2.5 (ZK composition)**.** *Let Π,Π' be dZKIP protocols (see Definition 2.2) that are run sequentially,where the secret shares of the private input to Π' can be efficiently computed from the shares of the private inputs and private outputs of Π (each verifier can compute its share of the private input to Π' on its own), and where the public inputs to Π' can be efficiently computed from the public inputs and outputs of Π. Suppose that:*

1. *Π is γ-dZK and α-complete. Π' is γ'-dZK.*
2. *If Π's inputs and outputs are in that protocol's input and output languages (L_{inp} and L_{out}), respectively, then they specify inputs for Π' that are in Π's input language L'_{inp}.*

Consider the composed protocol $(\Pi\circ\Pi')$, which runs Π, and then uses the resulting private output to specify inputs to an execution of Π'. Then the composed protocol $(\Pi\circ\Pi')$ satisfies $(\alpha+\gamma+\gamma')$-dZK.

2.3 An Example: Verifying Linear Equalities

We next provide an example for a distributed zero-knowledge protocol, which allows a prover to prove to two verifiers that some linear equality holds with respect to the shared inputs that the verifiers hold. The protocol is very simple and requires no interaction with the prover. Nevertheless, this protocol will be useful as a sub-protocol in our construction.

Example 1 (A protocol for linear equality)**.** ***Common inputs:*** *Field size $q\in\mathbb{N}$, Dimension $d\in\mathbb{N}$, coefficients $\alpha_1,\ldots,\alpha_d,\in\mathbb{GF}[q]$, a field element $z\in\mathbb{GF}[q]$.*

Secret-shared inputs: *A vector $X\in\mathbb{GF}[q]^d$, where each X_i is secret-shared as $[X_i]=(X_i^{(0)},X_i^{(1)})$. The prover knows the secret-shared values X and the shares $[X]$. The verifiers each know their own shares (respectively $X^{(0)}$ and $X^{(1)}$).*

Claim (to be verified): *$\sum_{i=1}^d\alpha_i\cdot X_i=z$. I.e. the input language is $L_{\mathsf{inp}}=\{((q,d,\{\alpha_i\},z),X):\sum_i\alpha_i\cdot X_i=z\}$, there are no private outputs (or output language).*

The Protocol: *For $j\in\{0,1\}$, verifier $\mathbf{V}_j$ (locally) computes $z_j=\sum_{i=1}^d\alpha_i\cdot X_i^{(j)}$ and sends z_j to $\mathbf{V}_{1-j}$. Both verifiers accept if $z=z_0+z_1 \pmod q$ and reject otherwise.*

Properties of the protocol: *This protocol offers perfect completeness and perfect soundness in the sense that the verifiers accept if and only if*

$$z=\sum_{i=1}^d\alpha_i\cdot(X_i^{(0)}+X_i^{(1)}) \pmod q.$$

To see that the protocol is perfect zero-knowledge, observe that for every input share $X^{(j)}$ for verifier $\mathbf{V}_j$, the simulator can compute $z_j=\sum_{i=1}^d\alpha_i\cdot X_i^{(j)} \pmod q$, and send to $\mathbf{V}_j$ the value $z-z_j \pmod q$ as its only message from $\mathbf{V}_{1-j}$.

3 Norm Verification

We build our protocol in multiple steps. Some of our sub-protocols output secret shares of values that are then checked in subsequent sub-protocols, e.g. checking that the secret-shared values are bits (i.e. 0 or 1).

In Section 3.1 we show a protocol that reduces checking that a secret-shared value is in some range to: (i) checking a linear equality over secret-shared values, and (ii) checking that shares generated in the protocol are secret shares of bits. In Section 3.2, we build on this to construct our main contribution: a protocol for verifying that the sum of secret-shared values is not larger than the field size (the sum is taken over the integers, not over the field). In particular, this protocol lets us reduce certifying a norm bound of secret-shared values to certifying quadratic constraints. We recall the appropriate form for quadratic constraint validation in Section 3.3 and combine these ingredients to derive our main result in Section 3.4.

3.1 Range-Check Subprotocol

We present a simple 1-message protocol that is helpful in verifying inequalities of the form $\sum_i \alpha_i Q_i \in [\beta_1, \beta_2] \pmod q$, where the coefficients $\alpha_i \in \mathbb{GF}[q]$ and the lower and upper bounds $\beta_1, \beta_2 \in \mathbb{GF}[q]$ are known and public (we view elements of $\mathbb{GF}[q]$ as integers in the set $\{-\lfloor q/2 \rfloor, \ldots, \lfloor q/2 \rfloor\}$), but the Q_i's are known only to the prover, not to the verifiers. In particular, the Q_i's are either secret-shared between the verifiers, or they are functions of secret-shared values (e.g. when we are verifying the inequality $\sum_i X_i^2 \leq B \pmod q$ over the secret-shared values X_i). The protocol reduces this verification task to: (i) the verification of an *equality* modulo q, an easier claim to deal with, and (ii) verifying that new secret-shared values (generated in the course of the protocol) are secret shares of bits.

Protocol overview. The prover computes $V = ((\sum_i \alpha_i Q_i) - \beta_1)$, and secret-shares the bits $\{v_j\}$ of V's binary representation. It also computes $U = (\beta_2 - (\sum_i \alpha_i Q_i))$ and secret-shares the bits $\{u_j\}$ of U's binary representation. Observe that $U, V \in [0, (\beta_2 - \beta_1)]$, and thus U and V can be represented using $b = \lceil \log(\beta_2 - \beta_1 + 1) \rceil$ bits. The verifiers get secret shares of these bits and verify that the sum of the values they represent is correct, i.e. that $(\sum_j v_j \cdot 2^j) + (\sum_j u_j \cdot 2^j) = (\beta_2 - \beta_1) \pmod q$ (recall that linear equalities over secret-shared values are easy for the verifiers to check). If this holds, and also: (i) the v_j's and the u_j's are all secret shares of bits, (ii) $\sum_j v_j \cdot 2^j = (\sum_i \alpha_i Q_i) - \beta_1$, and (iii) $q > 3(\beta_2 - \beta_1) + 2$, then indeed it must be the case that $\sum_i \alpha_i Q_i \in [\beta_1, \beta_2]$. Note that if Q_i's are themselves secret-shared values, then the verifiers can verify that condition (ii) holds, and all that remains is to verify that the secret-shared values are indeed of bits (but we will also use this protocol in situations where the verifiers do not have secret shares of the Q_i values). The protocol is in Figure 1, its complexity and guarantee are in Lemma 3.1.

Lemma 3.1. *The protocol of Figure 1 satisfies:*

1. ***Completeness:*** *If $\sum \alpha_i Q_i \in [\beta_1, \beta_2] \pmod q$ (this is the input condition), then the verifiers accept and it holds that: (i) $(\sum_{i=1}^n \alpha_i Q_i) - \beta_1 = \sum_{j=0}^{b-1} v_j \cdot 2^j \pmod q$, and (ii) the secret-shared values are bits: $\forall j : v_j, u_j \in \{0,1\}$ ((i) and (ii) are the output conditions).*

2. ***Soundness:*** *If $\sum_{i=1} \alpha_i Q_i \notin [\beta_1, \beta_2] \pmod q$, if the verifiers do not reject, then either: (i) $(\sum_{i=1}^n \alpha_i Q_i) - \beta_1 \neq \sum_{j=0}^{b-1} v_j \cdot 2^j \pmod q$, or (ii) for some j, either v_j or u_j is not in $\{0,1\}$.*

3. ***Zero-Knowledge****:The protocol is strong zero-knowledge as per Definition 2.2.*[4]

The proof consists of a single message of length $(2\lceil \log(\beta_2 - \beta_1 + 1) \rceil \cdot \log(q)$ from the prover to each verifier (this message contains the secret shares of the bits $\{v_j, u_j\}$). The prover performs $O(n + \log(\beta_2 - \beta_1 + 1))$ field operations. The verifiers each perform $O(\log(\beta_2 - \beta_1 + 1))$ field operations and communicate $\log(q)$ bits between themselves.

The proof of Lemma 3.1 is omitted: it follows from the construction. The simulator generates dummy shares of the bits v_j and u_j, and simulates receiving a message in the protocol that verifies the linear equality, as in Example 1.

Remark 3.2. *We note that if $(\beta_2 - \beta_1 + 1)$ is a power of two, we can simplify this protocol by skipping sending U and skipping the equality check. Indeed the fact that $V = (\sum_i \alpha_i Q_i) - \beta_1$ can be represented as $\sum_{j=0}^{b-1} v_j 2^j$, for bits v_j, is equivalent in this case to $V \in [0, 2^b - 1] = [0, \beta_2 - \beta_1]$.*

3.2 Detecting Wraparound

Our main technical contribution is a protocol that allows the servers to verify that the sum of the squares of the secret-shared values is not larger than the field size (where the sum is taken over the integers), i.e. that there is no "wraparound" when we take the sum of squares modulo the field size. The guarantees and complexity of the protocol are in Lemma 3.3. The protocol itself is in Figure 2.

In more detail: let q be the size of the field, constraints on which will be determined below. We view each X_i as an integer whose value is in the range $\{0, \ldots, q-1\}$. In what follows, all equalities and inequalities are over the integers unless we explicitly note otherwise (using and abusing the $\pmod q$ notation). The protocol considers a *promise problem*: in the YES case we have $\|X\|_2^2 \leq B$, whereas in the NO case we have $\|X\|_2^2 \geq q$ (i.e., there is wraparound). The verifiers accept or reject, and they also output secret shares $\{[w_j]\}$, which should be shares of bits (i.e. it should be true that $w_j \in \{0,1\}$, and this always holds in the YES case assuming the prover follows the protocol). Soundness guarantees that in the NO case, the probability that the verifiers accept *and* the output shares are all of bits is small. The verifiers will run a subsequent sub-protocol to verify that all w_j's are shares of bits.

Protocol overview. The verifiers pick a random vector $Z \in \{-1, 0, 1\}^d$, where each Z_i is -1 w.p. $1/4$, 1 w.p. $1/4$, and 0 w.p. $1/2$. The verifiers send Z to the prover, and verify that for a constant $\alpha > 0$ (set below, where we assume $q > 2\alpha\sqrt{B}$):

$$\sum_i Z_i X_i \in \left[-\alpha\sqrt{B}, \alpha\sqrt{B}\right] \pmod q. \tag{1}$$

This verification is performed using the protocol of Figure 1. We show that for an appropriate choice of α there is a small constant η s.t. in the YES case ($\|X\|_2^2 \leq B$) Equation (1) holds

[4] Technically, this protocol does not fall into our distributed verification model, as the verifiers do not hold secret shares of the Q_i's. Indeed, here the simulator can create its view without *any* access to shares of the Q_i's.

Protocol: Range Check $(\mathrm{mod}\ q)$

Common inputs: Field size $q \in \mathbb{N}$, number of variables $n \in \mathbb{N}$, coefficients $\alpha_1, \ldots, \alpha_n \in \mathbb{GF}[q]$ and claimed lower and upper bounds $\beta_1, \beta_2 \in \mathbb{GF}[q]$, viewed as integers in $\{-\lfloor q/2 \rfloor, \ldots, \lfloor q/2 \rfloor\}$, s.t. $\beta_1 \leq \beta_2$ and $q > 3(\beta_2 - \beta_1) + 2$.

Other inputs: The prover knows $Q_1, \ldots, Q_n \in \mathbb{GF}[q]$. We do not assume the verifiers have access to these Q_i's.

Secret-shared outputs: Shares $\{[v_j], [u_j]\}_{j=0}^{b-1}$, where $b = \lceil \log(\beta_2 - \beta_1 + 1) \rceil$ (for each j, each verifier outputs its respective shares, $(v_j^{(0)}, u_j^{(0)})$ or $(v_j^{(1)}, u_j^{(1)})$).

The Protocol:
The prover secret shares:

1. The b bits $(v_j)_{j \in [0,\ldots,b-1]}$ of $V = (\sum_i \alpha_i Q_i) - \beta_1 \pmod q$,
2. The b bits $(u_j)_{j \in [0,\ldots,b-1]}$ of $U = \beta_2 - (\sum_i \alpha_i Q_i) \pmod q$.

The verifiers verify the linear equality $(\sum_{j=0}^{b-1} v_j \cdot 2^j) + (\sum_{j=0}^{b-1} u_j \cdot 2^j) = \beta_2 - \beta_1 \pmod q$ (rejecting otherwise).

Figure 1: Range Check $(\mathrm{mod}\ q)$ Protocol

with probability at least $1 - \eta$ over the choice of Z. In the NO case, the probability is at most $1/2$. We can repeat the test in parallel to make the soundness error negligible. This effectively reduces our original problem to range-checks (and, via the protocol of Figure 1, to bit-checks). There is, however, a zero-knowledge issue: there's a non-negligible failure probability in the YES case, and the verifiers see whether the test failed or not (and the values Z that led to failure), which leaks information about X.

We could resolve this issue by increasing the field size to the point where the failure probability in the YES case becomes negligible, but this would entail a significant cost in the communication complexity required for sending field elements (we remark that increasing the field size doesn't reduce the soundness error). Instead, we take advantage of the fact that the protocol will be repeated many times (for soundness). The verifiers will not learn whether any of the individual repetitions succeeded, but only whether "many" of them succeeded. The threshold for "many" is set by a parameter τ indicating the fraction of repetitions that should succeed (by default we set $\tau = 3/4$, so the verifiers accept if and only if at least three quarters of the repetitions succeed). In the YES case this will happen with all but negligible probability (so we get statistical zero-knowledge). To accomplish this, we repeat the test r times, where in the k-th repetition the prover secret-shares a bit $g_k \in \{0,1\}$, indicating whether or not Equation (1) holds with respect to the k'th test. The verifiers multiply both sides of the equation by g_k, so that if $g_k = 0$ the test passes even though the equation doesn't hold. Later, the verifiers also verify that $\sum_k g_k \geq \tau \cdot r$ (the inequality can be verified in zero-knowledge using the protocol of Figure 1). Checking that Equation (1) holds after multiplying by g_k is done using the protocol of Section 3.3 for checking quadratic constraints over secret-shared values.

We make two minor modifications to improve the protocol's efficiency: First, the cost of the quadratic constraint protocol grows with the (square root of the) number of summands. Thus, for each iteration, each verifier computes (on its own) its secret share for the sum $\sum_i Z_i X_i$ (a linear function of the secret-shared X_i's). This gives the verifiers a single secret-shared field element that should be multiplied by g_k (instead of d). Indeed, in the full protocol, we fold further summands into this secret-shared sum. The second minor efficiency improvement we use is having the honest prover set *exactly* a $(1-\tau)$-fraction of the bits g_k to 0 (w.h.p. this means that some repetitions where Equation (1) holds will not be checked). This allows the verifiers to replace checking $\sum_k g_k \geq \tau \cdot r$ (an inequality) with a simpler equality check $\sum_k g_k = \tau \cdot r$.

Lemma 3.3. *Fix a bound B, a number of repetitions r, a desired completeness error for each repetition $\eta \in [0,1]$ and a threshold $\tau \in (1/2, 1]$ s.t. $\tau \cdot r$ is an integer. Let the field size q be at least $\max\{81B \cdot \ln(2/\eta), 100, 2r\}$.*

Let $\mathsf{Bin}(\ell; r, p)$ denote the probability that the Binomial distribution with parameters r and p has outcome (number of successes) at least ℓ. The protocol of Figure 2 has the following properties:

1. ***Completeness:*** *If $\sum X_i^2 \leq B$, then the probability that the prover aborts is at most:*

$$\rho_C = 1 - \mathsf{Bin}((\tau \cdot r); r, 1-\eta). \tag{2}$$

Thus, for $\tau \in (1/2, 1-\eta)$, we get that $\rho_C \leq \exp\left(-2(1-\eta-\tau)^2 \cdot r\right)$. For $\tau \in [1-\eta, 1]$, it is still the case that $\rho_C \leq r \cdot \eta$.

If the prover doesn't abort, then the verifiers accept and it holds that: (i) for every $k \in [r]$, $g_k \cdot S_k = 0$, and (ii) the output shares $\{[g_k], [v_{k,j}], [u_{k,j}]\}$ are shares of bits.

Wraparound Detection Protocol

Common inputs: Dimension $d \in \mathbb{N}$, claimed bound $B \in \mathbb{N}$, field size $q \in \mathbb{N}$, number of repetitions $r \in \mathbb{N}$, completeness error per repetition $\eta \in [0,1]$, threshold $\tau \in (1/2,1]$ for successful repetitions, s.t. $\tau \cdot r$ is an integer.

Secret-shared inputs: A vector $X \in \mathbb{GF}[q]^d$, where each X_i is secret-shared as $[X_i] = (X_i^{(0)}, X_i^{(1)})$.
The client (prover) knows the secret-shared values X and the shares $[X]$.
The servers (verifiers) each know their own shares (respectively $X^{(0)}$ and $X^{(1)}$).

Secret-shared outputs: $\{[g_k],[S_k],[v_{k,j}],[u_{k,j}]\}_{k\in[r],j\in[b]}$ where $b = \left(\lceil \log(2\alpha\sqrt{B}+1)\rceil\right)$.

The Protocol:
Fix $\alpha = \sqrt{\ln(2/\eta)}$. Let D_Z be the distribution that samples a d-dimensional vector where for each $i \in [1,\ldots,d]$, Z_i is drawn independently to be -1 w.p. $1/4$, 0 w.p. $1/2$ and 1 w.p. $1/4$

1. The following test is repeated in parallel r times, where in the k-th repetition:
   (a) The verifiers choose a random string $Z_k \sim D_Z$ and send it to the prover.
   The prover computes $Y_k = \sum_{i=1}^d Z_{k,i} X_i$.
   (b) If $Y_k \in [-\alpha\sqrt{B}, \alpha\sqrt{B}]$, the prover sets $g_k = 1$ and uses the protocol of Figure 1 to prove that $(\sum_i Z_{k,i} X_i) \in [-\alpha\sqrt{B}, \alpha\sqrt{B}]$. Let $\{v_{k,j}, u_{k,j}\}_{j\in[b]}$ be the secret-shared outputs of that protocol.
   Otherwise ($Y_k \notin [-\alpha\sqrt{B}, \alpha\sqrt{B}]$), the prover sets $g_k = 0$, sets the bits $\{v_{k,j}\}$ to be the bit representation of the value $2\alpha\sqrt{B}$, and sets the bits $\{u_{k,j}\}_{j\in[b]}$ to all be 0. Note that this setting ensures that the linear check in the protocol of Figure 1 (the last step of that protocol) succeeds.
2. If there are more than $(1-\tau)\cdot r$ repetitions in which $g_k = 0$ then the prover aborts (and the verifiers reject).
   Otherwise, the prover sets $g_k = 0$ for as many of the repetitions as needed to ensure there are *exactly* $(1-\tau)\cdot r$ such repetitions, and sends secret shares of $\{g_k\}_{k\in[r]}$.
3. For each $k \in [r]$, the verifiers will check that either $g_k = 0$ or the v_j's satisfy soundness condition (i) of the protocol of Figure 1 (see Lemma 3.1). Towards this, each verifier computes its a share of:
$$S_k = \left(\sum_{i=1}^d Z_{k,i} X_i\right) + \alpha\sqrt{B} - \left(\sum_{j\in[b]} 2^j \cdot v_{k,j}\right) \pmod q,$$
   Note that the sum S_k is a linear function of secret-shared values, so each verifier can indeed compute its share on its own. The protocol of Section 3.3 will later be used to check the quadratic equality $g_k \cdot S_k = 0 \pmod q$ (see the protocol of Figure 3).
4. The verifiers verify the linear equality $\sum_{k\in[r]} g_k = \tau \cdot r \pmod q$ (otherwise they reject).

Figure 2: Wraparound Detection Protocol

2. ***Soundness:*** *If $\sum X_i^2 \geq q$, the probability that the verifiers accept, and that (i) for every k it holds that $g_k \cdot S_k = 0$, and (ii) the shares $\{[g_k],[v_{k,j},[u_{k,j}]\}$ are all shares of bits, is at most*
$$\rho_S = \mathsf{Bin}((\tau\cdot r); r, \frac{1}{2}) \leq \exp\left(-2\left(\tau - \frac{1}{2}\right)^2 \cdot r\right). \quad (3)$$
3. ***Zero-Knowledge****: The protocol satisfies statistical zero-knowledge: the view of each verifier can be simulated up to statistical distance ρ_C (see Equation (2)).*

The protocol is public-coins, with 2 messages. The verifier's message is $(2\cdot d\cdot r)$ bits, the prover's response is of length $\left(\left(\left\lceil \log\left(\sqrt{2B\cdot\ln(2/\eta)}+1\right)\right\rceil + 2\right)\cdot r\cdot\log(q)\right)$. The prover performs $O\left((d+\log(B\cdot\ln(2/\eta)))\cdot r\right)$ field operations. The verifiers each perform $O\left((d+\log(B\cdot\ln(2/\eta)))\cdot r\right)$ field operations, and communicate $O(r\cdot\log(q))$ bits between themselves.

Proof. The protocol is in Figure 2, and the claimed complexity bounds follow by construction. We proceed to prove completeness for a single repetition (Proposition 3.4) and soundness for a single repetition (Proposition 3.5). For clarity, we omit the subscript k when we consider a single iteration. Let Y be the vector where $Y_i = Z_i X_i$. Recall that we set the parameter $\alpha = \sqrt{\ln(2/\eta)}$. The proof of the following completeness claim follows from standard subgaussian concentration, and deferred to the full version [ROCT23].

Proposition 3.4 (Completeness)**.** *Let $\sum_i X_i^2 \le B$. For every $\alpha > 0$ and every field size $q \ge 2\alpha\sqrt{B}$:*

$$\Pr_{Z_1,\ldots,Z_d}\left[Y \in [-\alpha\sqrt{B}, \alpha\sqrt{B}]\right] \ge 1 - 2e^{-\alpha^2}.$$

Proposition 3.5 (Soundness)**.** *For every B, every $\alpha \ge 1$ and every field size $q \ge \max\{81\alpha^2 B, 100\}$, if it is the case that $\sum_i X_i^2 > q$, then:*

$$\Pr_{Z_1,\ldots,Z_d}[Y \in [-\alpha\sqrt{B}, \alpha\sqrt{B}] \pmod q] \le \frac{1}{2}.$$

Proof. If any of the secret-shared w_j values are not bits, then the soundness condition is automatically satisfied. Thus, we only need to bound the probability that the verifiers accept conditioned on the event that all w_j's are bits. Under this conditioning, $(\sum_j w_j \cdot 2^j) \in [0, 2\alpha\sqrt{B}]$. The verifiers check that $(\sum_j w_j \cdot 2^j) = Y + \alpha\sqrt{B} \pmod q$, and thus they will reject unless $Y \in [-\alpha\sqrt{B}, \alpha\sqrt{B}] \pmod q$. We bound the probability that $Y \pmod q$ is in this range using a case analysis (soundness follows), where we view the value of each X_i modulo q as an integer between $-q/2$ and $q/2$.

Soundness, case *I* (large max). If:

$$\max_i |X_i| > 2\alpha\sqrt{B} \pmod q,$$

then let i^* be the argmax, where $|X_{i^*}| > 2\alpha\sqrt{B} \pmod q$ (note that this can only happen if $q > 4\alpha\sqrt{B}$). The probability, over the choice of Z, that $\sum_i Z_i X_i \pmod q$ lands in the interval $[-\alpha\sqrt{B}, \alpha\sqrt{B}]$ is at most $1/2$. To see this, fix all of Z's entries except the i^*-th entry. There are two cases:

- Let S be the sum of all Y_i's except the i^*-th $\pmod q$. If $S \in [-\alpha\sqrt{B}, \alpha\sqrt{B}]$, then when we add or subtract X_{i^*}, we end up outside the interval. This is because if $Z_{i^*} \neq 0$, then

  $$|Y_{i^*}| \in (2\alpha\sqrt{B}, q/2] \pmod q.$$

  Suppose w.l.o.g that Y_{i^*} is positive (modulo q), then:

  $$Y = S + Y_{i^*} \in (\alpha\sqrt{B}, q/2 + \alpha\sqrt{B}] \pmod q,$$

  and thus Y is outside the interval (recall that $q > 4\alpha\sqrt{B}$). A similar statement holds for the case that Y_{i^*} is negative (modulo q).

- If the sum of all Y_i's except the i^*-th is outside the interval, then w.p. $1/2$ we have that $Z_{i^*} = 0$ and the total sum is also outside the interval.

Soundness, case *II* (bounded max). Suppose it is the case that:

$$\max_i |X_i| \le 2\alpha\sqrt{B}.$$

In this case, we use the Berry-Esseen theorem to show that the distribution of Y is sufficiently close to a Normal distribution, and this allows us to bound the probability that its magnitude is small modulo q.

Theorem 3.6 (Berry-Esseen for our setting, see [Fel91; She10].)**.** *Suppose $\max_i |X_i| \le 2\alpha\sqrt{B}$, and take $\sigma^2 = \sum_i X_i^2/2$. Then for any $y \in \mathbb{R}$:*

$$\left|\Pr[Y \le y] - \Pr[\mathcal{N}(0,\sigma^2) \le y]\right| \le \frac{0.56 \cdot \alpha\sqrt{B}}{\sigma}.$$

A symmetric claim follows for bounding the probability that Y is above a value y'. For an interval $[y', y]$, the probability of Y lying in the interval can be bounded via a Union Bound on the probability that it is below y' and the probability that it is above y.

We use the Berry-Essen theorem, together with the concentration (and anti-concentration) properties of the Normal distribution, to bound the probability that Y's magnitude is small modulo q. In Claim 3.7 we bound the probability that Y lies in any single interval where its magnitude modulo q is small. In Claim 3.8 we bound the probability that Y is far from its expectation. To prove the soundness lemma, we combine these claims: bounding the probability that Y is not too far from its expectation and has small magnitude modulo q, and also that it is far from its expectation. We prove the following claims in the full version [ROCT23].

Claim 3.7. *Suppose that $\alpha \ge 1$, the field size is $q \ge 2\alpha\sqrt{B}$, and that $\max_i |X_i| \le 2\alpha\sqrt{B}$, and take $\sigma^2 = \sum_i X_i^2/2$. Then for every integer $u \ge 0$:*

$$\sum_{t=-u}^{u} \Pr\left[Y \in [-\alpha\sqrt{B} + t \cdot q, \alpha\sqrt{B} + t \cdot q]\right] \le (2u+1) \cdot \left(1.12 + \sqrt{\frac{2}{\pi}}\right) \cdot \frac{\alpha\sqrt{B}}{\sigma}$$

Claim 3.8. *Suppose that $\alpha \ge 1$, the field size is $q \ge 2\alpha\sqrt{B}$, and that $\max_i |X_i| \le 2\alpha\sqrt{B}$. Take $\sigma^2 = \sum_i X_i^2/2$. Then for every integer $t \ge 1$, it holds that:*

$$\Pr\left[|Y| \ge -\alpha\sqrt{B} + t \cdot q\right] \le \sqrt{\frac{2}{\pi}} \exp\left(-\frac{\left(-\alpha\sqrt{B} + t \cdot q\right)^2}{2\sigma^2}\right) + \frac{1.12\alpha\sqrt{B}}{\sigma}$$

As described above, putting these claims together, we can bound both the probability that $Y \pmod q$ has small magnitude but $|Y|$ isn't too large, and the remaining case, where $|Y|$ is large. Towards this, for a parameter $\delta > 0$ to be specified below, we set:

$$u = \left\lfloor \frac{(\sigma\sqrt{2\ln(1/\delta)}) + (\alpha\sqrt{B})}{q} \right\rfloor.$$

By a union bound over Claims 3.7 and 3.8, an easy computation (see the full verison [ROCT23]) establishes that

Claim 3.9. *Under the notation above,*

$$\Pr[Y \in [-\alpha\sqrt{B}, \alpha\sqrt{B}] \pmod q]$$
$$\leq \left(\alpha\sqrt{B} \cdot \left(1.12 + \sqrt{\frac{2}{\pi}}\right) \cdot \left(\frac{2\sqrt{2\ln(1/\delta)} + 0.04\sqrt{q}}{q}\right) + \delta\right).$$

Taking $\delta = e^{-3}$, and $q \geq 100$ s.t. $0.04\sqrt{q} + 2\sqrt{6} \leq 2.1\sqrt{q}$, we have that:

$$\Pr[Y \in [-\alpha\sqrt{B}, \alpha\sqrt{B}] \pmod q]$$
$$\leq \left(2.1 \cdot \left(1.12 + \sqrt{\frac{2}{\pi}}\right)\right) \frac{\alpha\sqrt{B}}{\sqrt{q}} + e^{-3}.$$

This probability can be bounded by $1/2$ for $q \geq \max\{81\alpha^2 B, 100\}$. □

In each repetition k, the prover can satisfy the conditions (i) $g_k \cdot S_k = 0$, and (ii) the secret-shared values $\{g_k, v_{k,j}, u_{k,j}\}$ are all shares of bits, if and only if either it holds that $Y_k \in [-\alpha\sqrt{B}, \alpha\sqrt{B}] \pmod q$, or the prover sets $g_k = 0$. Over all r repetitions, the prover can get the verifiers to accept while satisfying conditions (i) and (ii) above, if and only if in at least $\tau \cdot r$ of the repetitions, $Y_k \in [-\alpha\sqrt{B}, \alpha\sqrt{B}] \pmod q$.

Completeness and soundness over the r repetitions are thus bounded by claimed Binomial terms (see Equations (2) and (3)). The binomial tail terms can be further bounded by taking a Chernoff Bound. For completeness, in each repetition the probability of success is at least $1 - \eta$. The probability that in r repetitions we have fewer than $\tau \cdot r$ successes is bounded by:

$$\exp\left(-D_{\mathrm{KL}}\left(\tau \| 1-\eta\right) \cdot r\right) \leq \exp\left(-2\left(1-\eta-\tau\right)^2 \cdot r\right), \quad (4)$$

where $D_{\mathrm{KL}}(p\|q)$ is the KL divergence between the Bernoulli distribution with mean p and the distribution with mean q. For soundness, the probability of success in each repetition is at most $1/2$. The probability that in r repetitions we have at least $\tau \cdot r$ successes is bounded by:

$$\exp\left(-D_{\mathrm{KL}}\left(\tau \| 1/2\right) \cdot r\right) \leq \exp\left(-2\left(\tau - \frac{1}{2}\right)^2 \cdot r\right). \quad (5)$$

Remark 3.10. *We note that while the protocol is described as sending the vectors $Z_1, \ldots, Z_k$, standard techniques [SSS95; AS00] of using limited independence ($O(\log 1/\rho)$-wise independence suffices for the concentration bounds) can be used to reduce the communication to $O(r \log d \log 1/\rho)$ bits.*

Remark 3.11. *The analytic Chernoff bound expressions here can be rather loose for particular values of parameters (especially when η is close to zero). In practice, one can use better analytic estimates, or numerical estimates for concentration on binomial random variables We directly state the bounds in terms of the CDF of the Binomial distribution in Equations (2) and (3).*

Remark 3.12. *Suppose that for some L, H satisfying $-\alpha\sqrt{B} \leq L < 0 < H \leq \alpha\sqrt{B}$, we test for $Y \in [L, H]$ instead of $Y \in [-\alpha\sqrt{B}, \alpha\sqrt{B}]$. Then the soundness argument continues to hold as $Y \in [L, H] \Rightarrow Y \in [-\alpha\sqrt{B}, \alpha\sqrt{B}]$. Moreover, the completeness argument now holds with $\tau' = 2\exp(-\hat{\alpha}^2)$ where $\hat{\alpha} = \min(\frac{-L}{\sqrt{B}}, \frac{H}{\sqrt{B}})$. The protocol's practical efficiency can be improved by a careful choice of L, H, see Remark 3.2.*

Honest-verifier zero-knowledge. To prove zero-knowledge, we need to show that for a single verifier V_j, there exists a simulator S that does not know the private input X, but receives some share $X^{(j)}$ and generates a view that is statistically closely distributed to the view of the verifier V_j in a real interaction with the prover and the remaining verifier V_{1-j}. We sketch the construction of such a simulator. Let $\left\{X_1^{(j)}, \ldots X_d^{(j)}\right\}$ be the shares, given to V_j as input.

The Simulation: The simulator S starts by setting $\hat{X} = 0^d$ to be the (fake) input for the simulation and setting (fictitious) shares for V_{1-j} accordingly. Specifically, S sets the "shares" for the second verifier V_{1-j} to be $\hat{X}_i^{(1-j)} = -X_i^{(j)}$ and emulates an interaction between the two verifiers as follows.

1. The simulator S repeats the following r times in parallel, where in the k-th repetition:

   It simulates the verifier V_{1-j} in choosing a random string $Z_k \sim D_Z$ and receives from V_j its messages to the prover to determine Z_k.

   S simulates the prover, setting $Y_k = \sum_{i=1}^d Z_{k,i}\hat{X}_i$ to be 0. If $1 \leq k \leq \tau \cdot r$ the simulator sets $g_k = 1$, and otherwise sets $g_k = 0$.

   The simulator then sets the bits $\{v_{k,j}\}$ to be the Binary representation of the value $2\alpha\sqrt{B}$, and sets the bits $\{u_{k,j}\}_{j\in[b]}$ to all be 0. The simulator S simulates the view of $\mathbf{V}_j$ in the appropriate k executions of the protocol in Figure 1, with $\alpha_i = Z_{k,i}$.

2. The simulator selects shares for $\{g_k\}_{k\in[r]}$ and sends $\mathbf{V}_j$ its shares.

3. For each $k \in [r]$, the simulator computes $\mathbf{V}_j$'s share of S_k by taking the appropriate linear combination of existing shares:.

$$(s_k)^{(j)} = \left(\sum_{i=1}^{d} Z_{k,i}(X_i)^{(j)}\right) + \alpha\sqrt{B} - \left(\sum_{j\in[b]} 2^j \cdot (v_{k,j})^{(j)}\right) \pmod q$$

4. The simulator S emulates $\mathbf{V}_{1-j}$ in its interaction with $\mathbf{V}_j$ verifying the linear equality $\sum_{k\in[r]} g_k = \tau \cdot r \pmod q$ (see Example 1).

We first note that all choices of Z_k for all r repetitions are identically distributed in the simulation as in a real execution of the protocol. Condition on the event that for these choices, in the real execution of the protocol, it holds that there are at least τr values of k, for which $Y_k \in [-\alpha\sqrt{B}, \alpha\sqrt{B}]$. This means that the prover does not abort. Under the aforementioned conditioning, the view of $\mathbf{V}_j$ in the real execution is distributed identically to the view generated by the simulator. This is true since all that $\mathbf{V}_j$ ever sees are random shares and the views it sees in the simulation of the sub-protocol, which are perfectly simulated (since they are on accepting inputs). Finally, since in the real world, the probability that the prover aborts is at most ρ_C, we conclude that even without the conditioning, the two views are of statistical distance ρ_C. □

3.3 Verifying Quadratic Constraints

In this section we recall a protocol from the work of Boneh *et al.* [BBCGI19] for verifying a conjunction of low-degree constraints over secret-shared values. We focus on quadratic constraints. Let $\{X'_i\}_{i\in[n]}$ be a collection of secret-shared values. A quadratic constraint C_k is specified by public coefficients $\{c_{k,i,j} \in \mathbb{GF}[q]\}_{i,j\in\{0,\ldots,n\}}$ and a target value $a_k \in \mathbb{GF}[q]$, and the claim (to be verified) is that:

$$\sum_{i,j\in[0,n]} c_{k,i,j} \cdot X'_i \cdot X'_j = a_k,$$

where we use the convention that $X'_0 = 1$ (this allows us to include linear terms in the constraint).

Given a collection of m quadratic constraints as above, there is a 2-message protocol for verifying that all the constraints hold, where the communication complexity scales with the *square-root* of the number of variables. More generally, the communication complexity can be reduced to (roughly) $n^{1/r}$ by increasing the amount of interaction to $(2r-2)$ messages.

Verifying that a secret-shared value v is a bit can be expressed as the quadratic constraint $v^2 - v = 0$. Thus, we use this protocol to verify that (multiple) secret shares produced in other protocols are composed of bits.

Lemma 3.13 (Boneh *et al.* [BBCGI19, Corollary 4.7 and Remark 4.8])**.** *For a field of size q, an integer n that is a perfect square, a collection of secret-shared values $\{X'_i\}_{i=1}^n$, and quadratic constraints $C = \{c_{k,i,j}, a_k\}_{k\in\{1,\ldots,m\},i,j\in\{0,\ldots,n\}}$, there is a protocol for verifying the constraints as follows: all parties get as input the field size q and the constraints. The prover gets both shares of each value X'_i. Each verifier gets a single share of each secret-shared value, where:*

Completeness: *If all constraints hold, the verifiers accept.*

Soundness: *If at least one of the constraints is violated, then (no matter what strategy the prover follows) the probability that the verifiers accept is at most $\left(\frac{2\sqrt{n}}{q-\sqrt{n}} + \frac{m}{q}\right)$.*

Zero-Knowledge *: The protocol is perfect strong distributed honest-verifier zero-knowledge (see Definition 2.2).*

The protocol is public-coins, with 2 messages. The verifiers' message is $\log(q)$ bits and the prover's message is $((4\sqrt{n}+1)\cdot\log(q))$ bits. Let $\mathsf{nnz}(C)$ be the total number of non-zero coordinates in the constraints $\{c_{k,i,j}\}$. The prover performs $O(\mathsf{nnz}(C) + n\log(n) + m\log(m))$ field operations. In the course of verification, the verifiers each perform $O(\mathsf{nnz}(C) + n\log(n) + m\log(m))$ field operations, and they exchange $(2\sqrt{n}+2)\cdot\log(q)$ bits with each other.

Soundness amplification. The soundness error can be reduced by parallel repetition [BM88; Gol01]:

Corollary 3.14 (Parallel repetition of Lemma 3.13)**.** *For the same inputs considered in the protocol of lemma 3.13, repeating that protocol in parallel t times, where the verifiers accept iff all repetitions accept, gives a 2-message public-coins protocol with perfect completeness and zero-knowledge, where:*

1. *The soundness error is reduced to $\left(\frac{2\sqrt{n}}{q-\sqrt{n}} + \frac{m}{q}\right)^t$.*
2. *The communication and runtime complexities are t times larger than those in Lemma 3.13.*

3.4 Putting it Together: The Norm-Bound Protocol

Our final norm-bound protocol is described in Figure 3. We defer the proof to the full version [ROCT23].

Theorem 3.15. *Fix a bound B, parameters $r,t \in \mathbb{N}, \eta \in [0,1], \tau \in (1/2,1]$ s.t. $\tau \cdot r$ is an integer. Let the field size be $q \geq \max\{81B\cdot\ln(2/\eta), 1000, 3r\}$. The protocol of Figure 3 has the following properties:*

1. ***Completeness:*** *If the claim is true and the prover follows the protocol, the verifiers accept with prob. $\geq 1-\rho_C$, where*

$$\rho_C = 1 - \mathsf{Bin}((\tau\cdot r); r, 1-\eta),$$

L_2-Bound Protocol

Common inputs: Dimension $d \in \mathbb{N}$, claimed bound $B \in \mathbb{N}$, field size $q \in \mathbb{N}$, errors $\rho_C, \rho_S \in [0,1]$.

Secret-shared inputs: A vector $X \in \mathbb{GF}[q]^d$, where each X_i is secret-shared as $[X_i] = (X_i^{(0)}, X_i^{(1)})$.
The client (prover) knows the secret-shared values X and the shares $[X]$.
The servers (verifiers) each know their own shares (respectively $X^{(0)}$ and $X^{(1)}$).

Claim (to be verified): $\sum_{i=1}^{d} X_i^2 \leq B$ (where summation is over the integers).

The Protocol:

1. Run the wraparound protocol of Section 3.2, setting the parameters r, η, τ as in Theorem 3.15.
   The two messages in this subprotocol (the verifier sends the first message) are sent in messages 1 and 2 of the L_2-bound protocol. This results in secret shares of values $\{s_k\}_{k\in[r]}$ and alleged shares of bits $\{g_k, u_{k,j}, v_{k,j}\}$.
2. Run the protocol of Section 3.1 to verify that $\sum_i X_i^2 \in [0, B] \pmod q$.
   The prover's message in this sub-protocol is sent in message 2 of the L_2-bound protocol. Results in alleged shares of bits $\{v'_{j'}, u'_{j'}\}$.
3. Use the quadratic constraints protocol of Corollary 3.14, setting the number of repetitions t as in Theorem 3.15, to verify the following quadratic constraints:
   (a) $\sum_{i=1}^{d} X_i^2 = \sum_{j'} v'_{j'} \cdot 2^{j'} \pmod q$.
   (b) $\forall k \in [r], g_k \cdot s_k = 0$.
   (c) the secret-shared values $\{v'_{j'}, u'_{j'}\}$ and $\{g_k, u_{k,j}, v_{k,j}\}$ are all bits (i.e. in $\{0,1\}$).
   The two messages of this sub-protocol (the verifier sends the first message) are sent in messages 3 and 4 of the L_2-bound protocol.

If the verifiers in any of the sub-protocols executed above reject, then the verifiers in the L_2-bound protocol reject immediately. Otherwise, they accept.

Figure 3: L_2-Bound Protocol

where $\mathsf{Bin}(\ell; r, p)$ *denotes the probability that the Binomial distribution with parameters r and p has outcome (number of successes) at least* ℓ.

Thus, for $\tau \in (1/2, 1-\eta)$, $\rho_C \leq \exp\left(-2(1-\eta-\tau)^2 \cdot r\right)$. *For* $\tau \in [1-\eta, 1]$, $\rho_C \leq r \cdot \eta$.

2. ***Soundness:*** *If* $\sum X_i^2 > B$ *(over the integers), the probability that the verifiers accept is at most:*

$$\exp\left(-2\left(\tau - \frac{1}{2}\right)^2 \cdot r\right) + \left(\frac{2\sqrt{d + (\log(q)\cdot(r+2)/2)}}{q - 2\sqrt{d + (\log(q)\cdot(r+2)/2)}} + \frac{\log(q)\cdot(r+2)}{2q}\right)^t.$$

3. ***Zero-Knowledge****: The protocol satisfies statistical zero-knowledge: the view of each verifier can be simulated up to statistical distance* ρ_C *(see above).*

The protocol is public-coins, with 4 messages. The message lengths (in bits) are:

1. *the first messsage, sent by the verifier, is of length* $2dr$.
2. *the second message, sent by the prover, is of length at most* $(\frac{r}{2}+2)\cdot\log^2(q)$.
3. *The third message, sent by the verifier, is of length* $t \log q$.
4. *The fourth message, sent by the prover, is of length* $\left(t \cdot \left(4\sqrt{d + (\log(q)\cdot(r+2)/2)} + 1\right) \cdot \log(q)\right)$.

4 Differentially Private Secret Sharing

We first recall a notion of near-indistinguishability used in Differential Privacy, and the notion of differential zero knowledge from [Tal22]:

Definition 4.1 ((ε,δ)-closeness)**.** *Two random variables P and Q are said to be* (ε,δ)*-close, denoted by* $P \approx_{(\varepsilon,\delta)} Q$ *if for all events S, it holds that* $\Pr[P \in S] \leq e^{\varepsilon} \cdot \Pr[Q \in S] + \delta$*, and similarly,* $Pr[Q \in S] \leq e^{\varepsilon} \cdot \Pr[P \in S] + \delta$.

Definition 4.2 (Differential Zero Knowledge)**.** *We say a protocol* π *is* (ε,δ)*-Differentially Zero Knowledge w.r.t. L if there is an efficiently samplable distribution Q such that for all* $x \in L$*, the distribution* $\pi(x)$ *of the protocol's transcript on input x satisfies* $\pi(x) \approx_{(\varepsilon,\delta)} Q$.

In the full version [ROCT23] , we describe a way for a client to share a vector $X \in L = \{X \in \mathbb{Z}^d : \|X\|_2^2 \leq B\}$, while preserving differential zero knowledge. In brief, the client will share each X_i by adding (rounded) truncated Gaussian noise of magnitude large enough to guarantee differential privacy. The client (prover) will sample gaussian noise, truncated to have ℓ_2 norm at most Δ and take the ceiling to get R. It will then secret-share X as $-R$ and $X+R$. Since both X and R have bounded ℓ_2 norm, so do the secret shares. The verifiers check that the received secret shares have bounded ℓ_2 norm, which then implies a bound on the norm of the sum of any valid secret shares. For a large enough q, there can then be no wraparound. Validating the squared norm modulo q then yields :

Theorem 4.3. *Let* $(\varepsilon, \delta) \in (0,1)$ *and* $B \geq 1$. *Set* $q > 4\left(\sqrt{B}+\sqrt{d}+\sqrt{dBc_{\varepsilon,\frac{\delta}{2}}\cdot(1+\frac{2\sqrt{\log 8e/\delta}}{\sqrt{d}}+\frac{2\log 8e/\delta}{d})}\right)^2$. *Then the PINE differential ZK protocol (see the full version) has the following properties:*

Completeness: *If* $\sum_{i=1}^d X_i^2 \leq B$ *and the prover follows the protocol, the verifiers accept with probability* 1.

Soundness: *If* $\sum X_i^2 > B$ *(over the integers), the probability that the verifiers accept is at most:*

$$\left(\frac{2\sqrt{d+2\log(q)}}{q-2\sqrt{d+2\log(q)}}+\frac{2\log(q)+1}{q}\right)^t.$$

Zero-Knowledge: *The protocol satisfies* (ε,δ)*-differential zero knowledge: the view of each verifier can be efficiently simulated up to* (ε,δ)*-closeness.*

The protocol is public-coins, with 3 messages. The prover sends (in addition to the secret shares of x) $4\lceil log_2 q\rceil^2)$ *bits and* $\left(t\cdot\left(4\sqrt{d+2\lceil\log(B+1)\rceil}+1\right)\cdot\lceil\log_2 q\rceil\right)$ *bits in rounds 1 and 3 respectively. The verifiers send the second message of length* $(t\cdot\lceil\log_2 q\rceil)$ *bits.*

5 Performance Evaluation

We provide a further performance analysis for our protocols in different parameter regimes. As in Table 1, we analyze performance in terms of the communication overhead, beyond the communication that is needed to simply send secret shares for distributed aggregation (without any robustness to poisoning attacks). We consider aggregating d-dimensional integer vectors of ℓ_2 norm at most 2^{15} with $d \in \{10^4, 10^5, 10^6, 10^7\}$, where the aggregation is performed over secret-shared inputs in a field of size $q = 2^{64}$ or $q = 2^{128}$ (sending secret shares for the client's data requires $d \cdot \log q$ bits). The (statistical) zero-knowledge error is set to $\delta = 2^{-50}$ throughout.

In Table 3 we analyze PINE's communication overhead for a smaller soundness error 2^{-100}, fixing all other parameters to be the same as in the results of Table 1 (where the soundness error was 2^{-50}). The overhead for **Statistical ZK** is larger by a roughly 2x factor (compared with the results in Table 1): the larger overhead comes from additional repetitions of the sub-protocols. The overhead for **Differential ZK** is, similarly, larger by a roughly 2x factor (compared with the results in Table 1), except in the high-dimensional regime, where the main bottleneck is the increased field size, and the overhead for smaller soundness error is not much larger than the overhead for soundness error 2^{-50}.

In Table 4, we analyze PINE's communication overhead when the field is large ($q = 2^{128}$) for soundness errors 2^{-50} and 2^{-100}. For **Statistical ZK**, comparing with the performance for field size 2^{64}, the larger field size does not change the overhead for soundness error 2^{-50} (vs. Table 1), but the overhead is slightly smaller for soundness error 2^{-100} (vs. Table 3). For **Differential ZK**, the large field size reduces the overhead by a 2x or larger (for high dimensionality) factor (vs. Tables 1 and 3). This is because the main bottleneck in the Differential ZK protocol for high-dimensional data was having a large-enough field. For lower-dimensional data, the larger field size automatically gives a smaller soundness error, which reduces overheads. Indeed, for a field size this large, soundness 2^{-100} comes "for free", at no additional overhead (compared with the overhead for soundness error 2^{-50}).

In Table 5 we analyze **Differential PINE**'s performance as a function of the privacy parameter $\varepsilon \in \{0.1, 0.01, .0.001\}$. The overhead increases by (at most) a constant multiplicative factor for each order of magnitude improvement (reduction) in the privacy parameter. This is again due to the main "bottleneck" being the field size, which needs to grow linearly in $(1/\varepsilon)$ (thus the bit length of a field element grows with $\log(1/\varepsilon)$).

On the precision parameter. As discussed after the performance analysis in Section 1.1, we consider aggregating floating point vectors of Euclidean norm at most 1, and support $b = 15$ bits of precision. Our main focus is on distributed aggregation, where noise will be added to the aggregate before it is revealed to the servers (to guarantee differential privacy). Since we expect noise to be added to the aggregate, there is limited value in increasing the precision for individual contributions.

Further performance evaluation. We provide further evaluations in the full version [ROCT23]. First, we analyze PINE's *runtime* overhead for the prover and the verifier, and find that they are improved by 1-2 orders of magnitude compared to the prior work of [BBCGI19; ISR23]. We also provide more detailed evaluations of statistical PINE's communication overhead for many different choices of soundness and zero-knowledge errors.

	$d = 10^4$	$d = 10^5$	$d = 10^6$	$d = 10^7$
no robustness, # bits sent	$64 \cdot 10^4$	$64 \cdot 10^5$	$64 \cdot 10^6$	$64 \cdot 10^7$
prior work, overhead [BBCGI19; ISR23]	$> 1500\%$	$> 1500\%$	$> 1500\%$	$> 1500\%$
PINE, Statistical ZK, overhead	43.01%	6.27%	0.97%	0.26%
PINE, Differential ZK, overhead	8.92%	2.86%	0.63%	12.76%

Table 3: Communication analysis: our protocols and prior work. Parameters: **field size** $q \approx 2^{64}$ for aggregation, d-dimensional data, **soundness error** 2^{-100}, zero-knowledge error $\delta = 2^{-50}$. For differential ZK $\varepsilon = 0.1$.

	$d = 10^4$	$d = 10^5$	$d = 10^6$	$d = 10^7$
no robustness, # bits sent	$128 \cdot 10^4$	$128 \cdot 10^5$	$128 \cdot 10^6$	$128 \cdot 10^7$
prior work, overhead [BBCGI19; ISR23]	$> 1500\%$	$> 1500\%$	$> 1500\%$	$> 1500\%$
PINE, Statistical ZK, $\rho = 2^{-50}$, overhead	22%	3.18%	0.49%	0.13%
PINE, Statistical ZK, $\rho = 2^{-100}$, overhead	36%	4.58%	0.63%	0.15%
PINE, Differential ZK, $\rho = 2^{-50}$, overhead	4.77%	1.46%	0.32%	0.11%
PINE, Differential ZK, $\rho = 2^{-100}$, overhead	4.77%	1.46%	0.32%	0.11%

Table 4: Communication analysis: our protocols and prior work. Parameters: **field size** $q \approx 2^{128}$, d-dimensional data, **soundness error** $\rho = 2^{-50}$ **and** 2^{-100}, zero-knowledge error $\delta = 2^{-50}$. For differential ZK $\varepsilon = 0.1$.

	$d = 10^4$	$d = 10^5$	$d = 10^6$	$d = 10^7$
no robustness, # bits sent	$64 \cdot 10^4$	$64 \cdot 10^5$	$64 \cdot 10^6$	$64 \cdot 10^7$
prior work, overhead [BBCGI19; ISR23]	$> 1500\%$	$> 1500\%$	$> 1500\%$	$> 1500\%$
PINE, Differential ZK, $\varepsilon = 0.1$, overhead	4.77%	1.46%	0.32%	12.63%
PINE, Differential ZK, $\varepsilon = 0.01$, overhead	17.87%	14.14%	12.86%	25.14%
PINE, Differential ZK, $\varepsilon = 0.001$, overhead	17.87%	26.83%	25.40%	37.66%

Table 5: Communication analysis: **Differential ZK** protocol and prior work. Parameters: field size $q \approx 2^{64}$, d-dimensional data, soundness error $\rho = 2^{-50}$, zero-knowledge error $\delta = 2^{-50}$.

References

[AG21] Apple and Google. *Exposure Notification Privacy-preserving Analytics (ENPA) White Paper*. Available at https://covid19-static.cdn-apple.com/applications/covid19/current/static/contact-tracing/pdf/ENPA_White_Paper.pdf. 2021.

[AGJOP22] Surya Addanki, Kevin Garbe, Eli Jaffe, Rafail Ostrovsky, and Antigoni Polychroniadou. "Prio+: Privacy Preserving Aggregate Statistics via Boolean Shares". In: *Security and Cryptography for Networks - 13th International Conference, SCN 2022*. Ed. by Clemente Galdi and Stanislaw Jarecki. Vol. 13409. Lecture Notes in Computer Science. Springer, 2022, pp. 516–539. URL: https://doi.org/10.1007/978-3-031-14791-3%5C_23.

[AS00] Noga Alon and Joel H. Spencer. *The Probabilistic Method*. Second. Wiley-Interscience, 2000.

[BBCGI19] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. "Zero-Knowledge Proofs on Secret-Shared Data via Fully Linear PCPs". In: *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference*. Ed. by Alexandra Boldyreva and Daniele Micciancio. Vol. 11694. Lecture Notes in Computer Science. Springer, 2019, pp. 67–97.

[BBCGI21] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. "Lightweight Techniques for Private Heavy Hitters". In: *2021 IEEE Symposium on Security and Privacy (SP)*. 2021, pp. 762–776.

[BBCGI23] Dan Boneh, Elette Boyle, Henry Corrigan-Gibbs, Niv Gilboa, and Yuval Ishai. "Arithmetic Sketching". In: *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I*. Ed. by Helena Handschuh and Anna Lysyanskaya. Vol. 14081. Lecture Notes in Computer Science. Springer, 2023, pp. 171–202. URL: https://doi.org/10.1007/978-3-031-38557-5%5C_6.

[BBG19] Gilad Baruch, Moran Baruch, and Yoav Goldberg. "A Little Is Enough: Circumventing Defenses For Distributed Learning". In: *Advances in Neural Information Processing Systems 32: Annual Conference on*

Neural Information Processing Systems 2019. Ed. by Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett. 2019, pp. 8632–8642. URL: https://proceedings.neurips.cc/paper/2019/hash/ec1c59141046cd1866bbbcdfb6ae31d4-Abstract.html.

[BBGLR20] James Henry Bell, Kallista A. Bonawitz, Adrià Gascón, Tancrède Lepoint, and Mariana Raykova. *Secure Single-Server Aggregation with (Poly)Logarithmic Overhead*. 2020. URL: https://doi.org/10.1145/3372297.3417885.

[BGGKMRS22] James Bell, Adrià Gascón, Badih Ghazi, Ravi Kumar, Pasin Manurangsi, Mariana Raykova, and Phillipp Schoppmann. "Distributed, Private, Sparse Histograms in the Two-Server Model". In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. CCS '22. Association for Computing Machinery, 2022, pp. 307–321. ISBN: 9781450394505. URL: https://doi.org/10.1145/3548606.3559383.

[BIKMMPRSS17] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. "Practical Secure Aggregation for Privacy-Preserving Machine Learning". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. CCS '17. Association for Computing Machinery, 2017, pp. 1175–1191. ISBN: 9781450349468. URL: https://doi.org/10.1145/3133956.3133982.

[BM88] László Babai and Shlomo Moran. "Arthur-Merlin Games: A Randomized Proof System, and a Hierarchy of Complexity Classes". In: *J. Comput. Syst. Sci.* 36.2 (1988), pp. 254–276. URL: https://doi.org/10.1016/0022-0000(88)90028-1.

[CB17] Henry Corrigan-Gibbs and Dan Boneh. "Prio: Private, Robust, and Scalable Computation of Aggregate Statistics". In: *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017*. Ed. by Aditya Akella and Jon Howell. USENIX Association, 2017, pp. 259–282. URL: https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/corrigan-gibbs.

[CP22] Leo de Castro and Anitgoni Polychroniadou. "Lightweight, Maliciously Secure Verifiable Function Secret Sharing". In: *Advances in Cryptology – EUROCRYPT 2022*. Ed. by Orr Dunkelman and Stefan Dziembowski. Springer International Publishing, 2022, pp. 150–179. ISBN: 978-3-031-06944-4.

[Deh+23] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, Rodolphe Jenatton, Lucas Beyer, Michael Tschannen, Anurag Arnab, Xiao Wang, Carlos Riquelme Ruiz, Matthias Minderer, Joan Puigcerver, Utku Evci, Manoj Kumar, Sjoerd Van Steenkiste, Gamaleldin Fathy Elsayed, Aravindh Mahendran, Fisher Yu, Avital Oliver, Fantine Huot, Jasmijn Bastings, Mark Collier, Alexey A. Gritsenko, Vighnesh Birodkar, Cristina Nader Vasconcelos, Yi Tay, Thomas Mensink, Alexander Kolesnikov, Filip Pavetic, Dustin Tran, Thomas Kipf, Mario Lucic, Xiaohua Zhai, Daniel Keysers, Jeremiah J. Harmsen, and Neil Houlsby. "Scaling Vision Transformers to 22 Billion Parameters". In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 7480–7512. URL: https://proceedings.mlr.press/v202/dehghani23a.html.

[DKMMN06] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. "Our Data, Ourselves: Privacy Via Distributed Noise Generation". In: *Advances in Cryptology (EUROCRYPT 2006)*. Vol. 4004. Lecture Notes in Computer Science. Springer Verlag, 2006, pp. 486–503. URL: https://www.microsoft.com/en-us/research/publication/our-data-ourselves-privacy-via-distributed-noise-generation/.

[DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. "Calibrating Noise to Sensitivity in Private Data Analysis". In: *Theory of Cryptography*. Ed. by Shai Halevi and Tal Rabin. Springer Berlin Heidelberg, 2006, pp. 265–284. ISBN: 978-3-540-32732-5.

[Dos+21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=YicbFdNTTy.

[FCJG20] Minghong Fang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. "Local Model Poisoning Attacks to Byzantine-Robust Federated Learning". In: *29th USENIX Security Symposium, USENIX Security 2020*. Ed. by Srdjan Capkun and Franziska Roesner. USENIX Association, 2020, pp. 1605–1622. URL: https://www.usenix.org/conference/usenixsecurity20/presentation/fang.

[Fel91] William Feller. *An Introduction to Probability Theory and Its Applications, Volume 2*. Wiley, 1991.

[Gol01] Oded Goldreich. *The Foundations of Cryptography - Volume 1: Basic Techniques*. Cambridge University Press, 2001. ISBN: 0-521-79172-3. URL: http://www.wisdom.weizmann.ac.il/%5C%7Eoded/foc-vol1.html.

[GPRW23] Tim Geoghegan, Christopher Patton, Eric Rescorla, and Christopher A. Wood. *Distributed Aggregation Protocol for Privacy Preserving Measurement*. IETF Working Group Draft. https://datatracker.ietf.org/doc/draft-ietf-ppm-dap/. 2023. URL: https://datatracker.ietf.org/doc/draft-ietf-ppm-dap/.

[HKJ20] Lie He, Sai Praneeth Karimireddy, and Martin Jaggi. "Secure Byzantine-Robust Machine Learning". In: *CoRR* abs/2006.04747 (2020). arXiv: 2006.04747. URL: https://arxiv.org/abs/2006.04747.

[HLXCZ21] Meng Hao, Hongwei Li, Guowen Xu, Hanxiao Chen, and Tianwei Zhang. "Efficient, Private and Robust Federated Learning". In: *ACSAC '21: Annual Computer Security Applications Conference*. ACM, 2021, pp. 45–60. URL: https://doi.org/10.1145/3485832.3488014.

[HMR18] Robert Helmer, Anthony Miyaguchi, and Eric Rescorla. *Testing Privacy-Preserving Telemetry with Prio*. https://hacks.mozilla.org/2018/10/testing-privacy-preserving-telemetry-with-prio/. 2018.

[ISR23] ISRG. *DivviUp LibPrio Rust*. Retreived June 2023. 2023. URL: https://github.com/divviup/libprio-rs.

[Kai+21] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurelien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adria Gascon, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konecny, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrede Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Ozgur, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramer, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. *Advances and Open Problems in Federated Learning*. 2021. arXiv: 1912.04977 [cs.LG].

[LSTS20] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. "Federated Learning: Challenges, Methods, and Future Directions". In: *IEEE Signal Processing Magazine* 37.3 (2020), pp. 50–60.

[MM17] Saeed Mahloujifar and Mohammad Mahmoody. "Blockwise p-Tampering Attacks on Cryptographic Primitives, Extractors, and Learners". In: *Theory of Cryptography - 15th International Conference, TCC 2017*. Ed. by Yael Kalai and Leonid Reyzin. Vol. 10678. Lecture Notes in Computer Science. Springer, 2017, pp. 245–279. URL: https://doi.org/10.1007/978-3-319-70503-3%5C_8.

[PAFSTL23] Martin Pelikan, Sheikh Shams Azam, Vitaly Feldman, Jan "Honza" Silovsky, Kunal Talwar, and Tatiana Likhomanenko. *Federated Learning with Differential Privacy for End-to-End Speech Recognition*. 2023. arXiv: 2310.00098 [cs.LG].

[Pau+21] Matthias Paulik, Matt Seigel, Henry Mason, Dominic Telaar, Joris Kluivers, Rogier van Dalen, Chi Wai Lau, Luke Carlson, Filip Granqvist, Chris Vandevelde, Sudeep Agarwal, Julien Freudiger, Andrew Byde, Abhishek Bhowmick, Gaurav Kapoor, Si Beaumont, Áine Cahill, Dominic Hughes, Omid Javidbakht, Fei Dong, Rehan Rishi, and Stanley Hung. *Federated Evaluation and Tuning for On-Device Personalization: System Design and Applications*. 2021. arXiv: 2102.08503 [cs.LG].

[RMRB19] Swaroop Ramaswamy, Rajiv Mathews, Kanishka Rao, and Françoise Beaufays. *Federated Learning for Emoji Prediction in a Mobile Keyboard*. 2019. arXiv: 1906.04329 [cs.CL].

[RNFH19] Edo Roth, Daniel Noble, Brett Hemenway Falk, and Andreas Haeberlen. "Honeycrisp: Large-Scale Differentially Private Aggregation without a Trusted Core". In: *Proceedings of the 27th ACM Symposium on Operating Systems Principles*. SOSP '19. Association for Computing Machinery, 2019, pp. 196–210. ISBN: 9781450368735.

[ROCT23] Guy N. Rothblum, Eran Omri, Junye Chen, and Kunal Talwar. *PINE: Efficient Norm-Bound Verification for Secret-Shared Vectors*. 2023. arXiv: 2311.10237 [cs.CR].

[RSWP22] Mayank Rathee, Conghao Shen, Sameer Wagh, and Raluca Ada Popa. "ELSA: Secure Aggregation for Federated Learning with Malicious Actors". In: *IACR Cryptol. ePrint Arch.* (2022), p. 1695. URL: https://eprint.iacr.org/2022/1695.

[RU23] Olivia Röhrig and Maxim Urschumzew. *dpsa4fl: Differential Privacy for Federated Machine Learning with PRIO*. 2023. URL: https://github.com/dpsa4fl/overview.

[RZHP20] Edo Roth, Hengchu Zhang, Andreas Haeberlen, and Benjamin C. Pierce. "Orchard: Differentially Private Analytics at Scale". In: *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. USENIX Association, 2020, pp. 1065–1081. ISBN: 978-1-939133-19-9.

[SGA20] Jinhyun So, Basak Guler, and A. Salman Avestimehr. *Turbo-Aggregate: Breaking the Quadratic Aggregation Barrier in Secure Federated Learning*. 2020. arXiv: 2002.04156 [cs.LG].

[SGA21] Jinhyun So, Basak Güler, and A. Salman Avestimehr. "Turbo-Aggregate: Breaking the Quadratic Aggregation Barrier in Secure Federated Learning". In: *IEEE Journal on Selected Areas in Information Theory* 2.1 (2021), pp. 479–489.

[SH21] Virat Shejwalkar and Amir Houmansadr. "Manipulating the Byzantine: Optimizing Model Poisoning Attacks and Defenses for Federated Learning". In: *28th Annual Network and Distributed System Security Symposium, NDSS 2021*. The Internet Society, 2021. URL: https://www.ndss-symposium.org/ndss-paper/manipulating-the-byzantine-optimizing-model-poisoning-attacks-and-defenses-for-federated-learning/.

[She10] I.G. Shevtsova. "An improvement of convergence rate estimates in the Lyapunov theorem". In: *Dokl. Math.* 82 (2010), pp. 862–864. URL: https://doi.org/10.1134/S1064562410060062.

[SHKR22] Virat Shejwalkar, Amir Houmansadr, Peter Kairouz, and Daniel Ramage. "Back to the Drawing Board: A Critical Evaluation of Poisoning Attacks on Production Federated Learning". In: *43rd IEEE Symposium on Security and Privacy, SP 2022*. IEEE, 2022, pp. 1354–1371. URL: https://doi.org/10.1109/SP46214.2022.9833647.

[SKSM19] Ziteng Sun, Peter Kairouz, Ananda Theertha Suresh, and H. Brendan McMahan. "Can You Really Backdoor Federated Learning?" In: *CoRR* abs/1911.07963 (2019). arXiv: 1911.07963. URL: http://arxiv.org/abs/1911.07963.

[SSS95] Jeanette P. Schmidt, Alan Siegel, and Aravind Srinivasan. "Chernoff–Hoeffding Bounds for Applications with Limited Independence". In: *SIAM Journal on Discrete Mathematics* 8.2 (1995), pp. 223–250.

[Tal+23] Kunal Talwar, Shan Wang, Audra McMillan, Vojta Jina, Vitaly Feldman, Bailey Basile, Áine Cahill, Yi Sheng Chan, Mike Chatzidakis, Junye Chen, Oliver Chick, Mona Chitnis, Suman Ganta, Yusuf Goren, Filip Granqvist, Kristine Guo, Frederic Jacobs, Omid Javidbakht, Albert Liu, Richard Low, Dan Mascenik, Steve Myers, David Park, Wonhee Park, Gianni Parsa, Tommy Pauly, Christian Priebe, Rehan Rishi, Guy Rothblum, Michael Scaria, Linmao Song, Congzheng Song, Karl Tarbe, Sebastian Vogt, Luke Winstrom, and Shundong Zhou. "Samplable Anonymous Aggregation for Private Federated Data Analysis". In: *CoRR* abs/2307.15017 (2023). arXiv: 2307.15017. URL: https://doi.org/10.48550/arXiv.2307.15017.

[Tal22] Kunal Talwar. "Differential Secrecy for Distributed Data and Applications to Robust Differentially Secure Vector Summation". In: *3rd Symposium on Foundations of Responsible Computing, FORC 2022*. Ed. by L. Elisa Celis. Vol. 218. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 7:1–7:16.

[Xu+23] Mingbin Xu, Congzheng Song, Ye Tian, Neha Agrawal, Filip Granqvist, Rogier van Dalen, Xiao Zhang, Arturo Argueta, Shiyi Han, Yaqiao Deng, Leo Liu, Anmol Walia, and Alex Jin. "Training Large-Vocabulary Neural Language Models by Private Federated Learning for Resource-Constrained Devices". In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2023, pp. 1–5.

[XZACKMRZ23] Zheng Xu, Yanxiang Zhang, Galen Andrew, Christopher Choquette, Peter Kairouz, Brendan Mcmahan, Jesse Rosenstock, and Yuanbo Zhang. "Federated Learning of Gboard Language Models with Differential Privacy". In: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*. Ed. by Sunayana Sitaram, Beata Beigman Klebanov, and Jason D Williams. Association for Computational Linguistics, 2023, pp. 629–639. URL: https://aclanthology.org/2023.acl-industry.60.

[ZLLBRZGS23] Shuangfei Zhai, Tatiana Likhomanenko, Etai Littwin, Dan Busbridge, Jason Ramapuram, Yizhe Zhang, Jiatao Gu, and Joshua M. Susskind. "Stabilizing Transformer Training by Preventing Attention Entropy Collapse". In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, 2023, pp. 40770–40803. URL: https://proceedings.mlr.press/v202/zhai23a.html.