Practical Private Computation and Zero-Knowledge Tools for Privacy-Preserving Distributed Data Mining*

Yitao Duan John Canny

Abstract

In this paper we explore private computation built on vector addition and its applications in privacy-preserving data mining. Vector addition is a surprisingly general tool for implementing many algorithms prevalent in distributed data mining. Examples include linear algorithms like voting and summation, as well as non-linear algorithms such as SVD, PCA, k-means, ID3, machine learning algorithms based on Expectation Maximization (EM), etc., and all algorithms in the statistical query model [27]. The non-linear algorithms aggregate data only in certain steps, such as conjugate gradient, which are *linear* in the data. We introduce a new and highly efficient VSS (Verifiable Secret-Sharing) protocol in a special but widely-applicable model that allows secret-shared arithmetic operations in such aggregation steps to be done over small fields (e.g. 32 or 64 bits). There are two major advantages: (1) in this framework private arithmetic operations have the same cost as normal arithmetic and (2) the scheme admits extremely efficient zero-knowledge (ZK) protocols for verifying properties of user data. As a concrete example, we present a very efficient zero-knowledge method based on random projection for verification that uses a linear number of inexpensive *small field* operations, and only a logarithmic number of large-field (1024 bits or more) cryptographic operations. Our implementation shows that the approach can achieve orders of magnitude reduction in running time over standard techniques (from hours to seconds) for large scale problems. The ZK tools provide efficient mechanisms for dealing with actively cheating users, a realistic threat in distributed data mining which has been lacking practical solutions.

1 Introduction

Many kinds of analysis depend on mining data from a group of users. Examples include linear algorithms like voting and summation, as well as non-linear ones such as regression, SVD, k-means, ID3, and many machine learning algorithms based on the EM (Expectation Maximization). These analysis methods can be found in a wide range of applications from E-commerce to medical research. We consider a fully distributed setting where each user holds her own data and participates in mining tasks managed by a few data miners. One example scenario could be e.g. two research institutes analyzing user survey data. In all these cases, it is very important to protect user privacy to the maximum extent possible. Indeed in some situations laws even prohibit service (e.g. health care) providers from disclosing customers information, even to other providers. At the same time, it is important to perform the same computation with no loss of accuracy (which can mean lost revenue for the provider, or misleading results for research institutes). Providers do not benefit directly from privacy technology, so the costs to them must be as small as possible, and ideally zero. These are the constraints that have guided the work reported here.

Privacy-preserving data mining has been an active area of research since it was introduced by Agrawal and Srikant [1] and Lindell and Pinkas [28]. Existing solutions use either randomization (e.g. [16, 12]) or cryptographic techniques (e.g. [28, 11, 34, 36, 35]) to protect privacy. Besides sacrificing accuracy, randomization has been shown to provide very little privacy protection in many cases unless sufficient amount of noise is introduced [26, 2, 10]. Works in the second category typically make use of private computation protocols and enjoy provable privacy provided by the cryptographic primitives. However, there has been two major inadequacies with almost all existing cryptography-based solutions: (1) the schemes are not practical for large scale systems due to their heavy use of expensive cryptographic operations, and (2) there is generally no efficient mechanism to handle actively cheating users. All the works in [28, 11, 34, 36, 35] deal with only *passive* adversary who never deviates from specified behavior. The assumption is clearly not realistic when the data comes from individual users, some of whom may incentivized to bias the computation or have their machines corrupted by hackers. Securing the system against such threat is a necessary step in making privacy-preserving data min-

*Both authors are with Computer Science Division, University of California, Berkeley, CA 94720, USA. Emails: {duan, jfc}@cs.berkeley.edu.

ing into real-world applications.

Regarding the first issue, we note that cryptography provides primitives with various level of efficiency [7, 18, 8, 3]. While both addition and multiplication are possible in most of these schemes, the practical overhead for multiplication is much higher. Thus one way to build a practical private computation scheme is to avoid the multiplication and focus on algorithms that can be implemented using addition-only steps. Surprisingly, a lot can be done with such an approach. The standard algorithms for many of the above-mentioned analysis use gradient steps which sum vector data from the users. These steps are *linear* in per user data and can be implemented using an addition-only approach. Implementing the algorithms using summation forms has been used by other works as a general approach to parallelize the algorithms and run them in a distributed fashion. For example, [6] and [9] showed that many popular algorithms has a summation implementation that can be computed with Google's MapReduce framework, which is a distributed programming construct over clusters that is being successfully deployed in production. The examples included an EM algorithm for pLSI [24], Locally Weighted Linear Regression (LWLR), Naive Bayes (NB), PCA, etc. In fact all the algorithms in the statistical query model [27] can be expressed in this form. They demonstrated the versatility of vector-addition in implementing statistical learning algorithms. This phenomena has also been observed by researchers in privacy technology field and used as a way to implement the algorithms with privacy. These works include private SVD [3], EM-based collaborative filtering [4], and link analysis such as the HITS algorithm [14].

Existing private addition-only solutions are still not always practical due to their heavy use of public-key operations for information hiding and/or verification. While these examples have the same asymptotic complexity as the standard algorithms for those problems, the constant factors imposed by public-key operations are prohibitive for large scale systems. On a typical computer today there is a six order of magnitude difference between the crypto exponentiations (in large field) needed for secure homomorphic computation (order of milliseconds) and regular arithmetic operations in small (32- or 64-bit) fields (fraction of a nano-second). Both homomorphic arithmetic [8, 3] and VSS (Verifiable Secret Sharing) [5] rely on public-key operations for verification. Even fast-track VSS [18] does not reduce the asymptotic number of crypto operations.

1.1 Our Results We present a novel private vector addition protocol in a special but widely-applicable model (to be elaborated later) based on secret sharing over *small* field, which avoids the manipulation of large integers so that private computation on each server has the *same* cost as regular non-private computation. To address the issue of cheating users, we present a very efficient probabilistic zero-knowledge protocol that verifies that the L2-norm of user vector is bounded by a constant. This is to prevent a malicious user from exerting too much influence on the computation. The protocol uses a linear number of *small-field* operations, and a *logarithmic* number of large field crypto operations (in the size of the user data). This improves significantly over standard techniques that require at least linear number of such expensive steps. In our scheme, the cost will be dominated by the linear number of small-field operations that one has to pay even when the computation is done directly on user data without privacy. In a sense, privacy is almost "free" even counting constant factors. Experimental results with real implementation show that it is extremely fast for typical data (order of seconds for a million-element vector).

Our basic approach, VSS-based private computation over small field and random projection-based method for verifying (in zero-knowledge) high-dimensional data, also opens doors to many other practical ZK tools. These include a zero-knowledge vector equality test [13] and potentially many others. These protocols also use only a small (constant or logarithm in the size of input) number of large field operations so that they preserve the efficiency we obtained by using small field VSS. Such tools provide practical means to deal with realistic adversary models which accommodate active cheating from the users.

2 Preliminaries

We consider the scenario where a small number $\kappa > 1$ of servers belonging to different service providers or institutes collaboratively mining data collected from n users. We define a *server* as all the computation units under the control of a single entity. It can be a cluster of thousands of machines so that it has the capability to support a large number of users. From information-sharing perspective, it suffices to view all the machines under a single control as a single entity which is referred to as a server in this paper.

This is different from the models many privacy-preserving data mining schemes consider (e.g. [28, 11, 34, 36, 35])). In their models, each server holds either a subset of attributes of all users, or all attributes of a subset of users. The former is called vertical partition [33, 34] and the latter horizontal partition [25, 28]. Our model generalizes the horizontal partition model. In our model, no user data ever leaves its owner unprotected and no server can obtain any information about user

data other than what can be inferred from the final results as long as at least one server is uncorrupted. Arguably, this offers much better privacy protection since it conforms to the "user-owned and operated" privacy principle that is both natural and effective in many situations [3].

Threat Model *Let $\alpha \in [0, 1)$ be the upper bound on the fraction of the dishonest users in the system.* [1] *We consider a computationally bounded adversary whose capability of corrupting parties is modelled as follows:*

1. *An adversary may actively corrupt at most $\lfloor \alpha n \rfloor$ users, taking full control of their machines and causing them to deviate arbitrarily from the specified protocol.*
2. *In addition to 1, we also allow the same adversary to passively corrupt $\kappa - 1$ server(s). When a party is passively corrupted, all her data is exposed to the adversary but the party continues to follow the protocol.*

This threat model is similar to that of [17] in that some of the participants are actively corrupted while some others are passively corrupted by the same adversary *at the same time*. This is an extension to the general adversary structure introduced by Hirt and Maurer [22, 23]. Our model does not satisfy the feasibility requirements of [22, 23] and [17]. We avoid the impossibility by considering addition only computation.

The model models realistic threats in our target applications. In general, users are not trustworthy. Some may be incentivized to bias the computation (e.g. to drive down/up the price of an item), some may have their machines corrupted by hackers. So we model them as active adversaries and our protocol ensures that the active cheating from a small number of users will not exert large influence on the computation. This greatly improved over existing privacy-preserving data mining solutions (e.g. [28, 36, 35, 34]) which handle only purely passive adversary. The corrupted servers, on the other hand, are modeled as passive adversaries that can share data among themselves and with corrupted users. This is appropriate for many data mining applications where the servers are cooperative but belong to separate, non-colluding entities. One example is the case where two hospitals mine data collected from patients. In this context service providers benefit from accurate computation so they do not have the incentive to disrupt or bias it. Their threat is therefore mainly to users privacy thus passive.

2.1 The Computation Let ϕ be a small integer (e.g. 32- or 64-bit). Since we need signed values, we consider the specific coset representatives of the integers mod ϕ in the range $-\lfloor \phi/2 \rfloor, \ldots, \lfloor \phi/2 \rfloor$ if ϕ odd, or $-\lfloor \phi/2 \rfloor, \ldots, \lfloor \phi/2 \rfloor - 1$ if ϕ even. We write $\mathbb{Z}_\phi$ for this additive group. [2]

Let a_i be private user data for user i and A be public information. Both can be matrices of arbitrary dimensions with elements from arbitrary domains. Our scheme supports any iterative algorithms whose $(t+1)$-th update can be expressed as

$$A^{(t+1)} = F(\sum_{i=1}^{n} d_i^{(t)}, A^{(t)})$$

where $d_i^{(t)} = G(a_i, A^{(t)}) \in \mathbb{Z}_\phi^m$ is an m-dimensional data vector for user i computed locally, and $A^{(t+1)}$ is the $(t+1)$-th update to A. Typical values for both m and n can range from thousands to millions. Both functions F and G are in general non-linear.

As mentioned before, this is a powerful model that includes a large number of popular data mining and machine learning algorithms. Additional examples include SVD, k-means, ID3, etc., most gradient-based and EM algorithms, and all the algorithms in the statistical query model [27]. Also see [3, 4, 14] for more examples.

If $d_i^{(t)}$ are computed locally by each user, our protocol allows calculation of the sum, and thence the next update $A^{(t+1)}$ without disclosing any information about $d_i^{(t)}$ or a_i. In the following we only describe the protocol for one such iteration since the entire algorithm is simply a sequential invocations of the same protocol. The superscript is thus dropped from the notation.

2.2 Leakage by the Sums Our private summation protocol guarantees that no more information beyond the sums is revealed. One thus must be careful about the potential leakage caused by the sums. For this we draw on the work in statistical database privacy [2, 10, 15]. Roughly speaking, these works showed that releasing the sums with sufficient amount of additive random noise could guarantee strong privacy provided that Tm is bounded where T is the total number of iterations. In our settings, we are able to prove that, when

[1] Most mining algorithms need to bound the amount noise in the data to produce meaningful results. This means that the fraction of cheating users must be below a reasonable threshold (e.g. $\alpha < 20\%$). Under this condition our scheme should give the honest players fairly good privacy protection.

[2] Since our computation involves addition only, there is no requirement that this group be a field. So ϕ need not be prime, and indeed it simplifies computation to take $\phi = 2^{32}$ or 2^{64} i.e. word-length or long integers.

n is large, external noise is *not* necessary for maintaining privacy. Instead, the randomness associated with an adversary's inherent uncertainty about unknown data is enough to provide the same level of protection as achieved with additive noise [2, 10, 15] but with much higher precision. We will give this topic a rigorous treatment in another paper. However, the soundness of the approach introduced in *this* paper does not have to depend on the new results which are yet to be established: it will be clear that it is trivial to introduce additive noise in our framework so the privacy can be guaranteed by well-established work in [2, 10, 15]. And in the event that noise is deemed necessary, the ZK tool that we introduce in this paper becomes even more crucial: the system needs to bound the magnitude of user-introduced noise.

As an additional indication that the sums are benign to privacy, notice that for some algorithms such as SVD, etc. [31, 3], the sums can be approximated from the final result so they do not leak more information at all.

2.3 Security Properties Since all user vectors are hidden, it is necessary to impose checkable bounds on the user data. Otherwise a single malicious user could corrupt the computation with values as large as $\lfloor \phi/2 \rfloor$ and not be discovered. For this purpose, we use a bound on the L2-norm of each user vector. This is both computationally natural for many applications, and also supports a very efficient, randomized check. The maximum L2-norm for a user vector is defined to be L, which implies that every component of the user vector must be in the range $[-L, L]$.

Note that L must be substantially less than ϕ. First of all, since n user vectors are added to reach a final total mod ϕ, each component value should be less than $1/n$ times $\phi/2$. Secondly, L should be much smaller than ϕ to ensure low probability of modular arithmetic anomalies (this is made precise in theorem 3.1).

Our protocol achieves the following which are similar to those introduced in [3]:

1. **Privacy**: For any honest user i who follows the protocol and inputs valid data, no participants, except herself, should gain any information about d_i, except what is implied by the final aggregate and the validity property below.
2. **Validity**: A user vector d_i that is included in the computation must satisfy that, with high probability, $|d_i|_2 < L$ where $|d_i|_2$ denotes the L2-norm of the vector d_i.
3. **Correctness**: The computation should produce the correct sum of all *valid* users data.

We adopt the privacy definition of [19] (Chapter 7) and prove the privacy of our protocol in a simulation paradigm that is common in numerous cryptography works and can be traced back to the notion of zero-knowledge [21]. Informally, a protocol is private if, for any adversary that corrupts a subset of the participants as allowed by the protocol, there exists a feasible simulator that, given the corrupted parties data and the final result, can generate a view that, to the adversary, is indistinguishable from the transcript of a real execution of the protocol. This guarantees that whatever information the adversary can obtain after attacking the protocol can be actually generated by himself (by running the simulator) thus no more information about honest parties data is leaked. For formal definition please see [19] (Chapter 7).

The privacy our protocol achieves is information-theoretic, i.e. it holds against an adversary with unbounded computation power, as is provided by the secret sharing and Pedersen commitment scheme we use [30, 7]. However, the success of the validity check relies on some standard assumptions (e.g. DDH or discrete log) so is only computational.

For validity, we provide an extremely efficient zero-knowledge protocol that, instead of verifying $|d_i|_2$ directly, checks the square sum of the vector's projections on some random directions. We show in theorem 3.1 that this check is effective in bounding $|d_i|_2$. The protocol is probabilistic and has a small failure probability. In a particular failure mode (i.e. false rejection), the protocol leaks one bit of information about user data, i.e. at least one of the projections is large (but it does not allow one to infer which direction(s)). This is made precise in theorem 3.2.

2.4 Bounding the L2-Norm Bounding the L2-norm of a user's vector is a natural and effective way to restrict the amount malicious influence on the computation a cheating user could cause. This can be shown from several perspectives. Firstly, notice that the result of the computation depends on the sums of n vectors. To drive the sums away from correct positions by a large amount, a malicious user must input a vector with sufficient "length", which is naturally measured by its L2-norm. This is especially evident for algorithms whose results are simply the vector sums (e.g. k-means). In this case even the precision of the final result is often measured by the L2-norm of the error vector (see e.g. [2]), which, by triangle inequality, is bounded by the sum of the L2-norms of all noise vectors.

Secondly, many perturbation theories measure the perturbation to the system in terms of various forms of (matrix and vector) norms, many of which can be

easily transformed into vector L2-morns. For example, let $\tilde{\cdot}$ denote the perturbed quantity and σ_i the i-th singular value of a matrix A, the classical Weyl and Mirsky theorems [32] bound the perturbation to A's singular values in terms of the spectral norm $\|\cdot\|_2$ and the Frobenius norm $\|\cdot\|_F$ of $E := A - \tilde{A}$, respectively:

$$\max_i |\tilde{\sigma}_i - \sigma_i| \leq \|E\|_2 \text{ and } \sqrt{\sum_i (\tilde{\sigma}_i - \sigma_i)^2} \leq \|E\|_F$$

The spectral norm can be bounded from above by Frobenius norm: $\|E\|_2 \leq \|E\|_F$. And if each row, denoted a_i, of the matrix A is held by a user, the Frobenius norm of the matrix E can be expressed in terms of vector L2-norms:

$$\|E\|_F = \sqrt{\sum_{i=1}^{n} |\tilde{a}_i - a_i|_2^2}$$

Clearly bounding the vector L2-norm provides an effective way to bound the perturbation of the results. Similar techniques was also used in e.g. [14].

And finally, bounding the L2-norm can also be the basis of other, more specific checks. For instance, in a voting application, the protocol can be used with $L = 1$ to ensure that each user only exercises one vote.

3 Practical VSS-Based Vector Addition

For simplicity, we only describe the protocol for the case of $\kappa = 2$. It is straightforward to extend it to support $\kappa > 2$ servers (by substituting the $(2, 2)$-threshold secret sharing scheme with a (κ, κ) one). Using more servers strengthens the privacy protection but also incurs additional cost. We do not expect the scheme will be used with a large number of servers.

3.1 Basic Computation Let Q be the initial set of qualified users. Let T_1 and T_2 denote the two servers. The basic computation is carried out as follows:

1. User $i \in Q$ generates a uniformly random vector $u_i \in \mathbb{Z}_\phi^m$ and computes $v_i = d_i - u_i \mod \phi$. She sends u_i to T_1 and v_i to T_2.
2. User i gives a ZK proof to both servers that her input is valid using the protocol that will be described in Section 3.2. If she fails to do so, both servers exclude her from Q.
3. If enough (e.g. more than 80% of all users in the group) inputs are collected and pass the validation test, T_1 computes $\mu = \sum_{i \in Q} u_i \mod \phi$ and T_2 computes $\nu = \sum_{i \in Q} v_i \mod \phi$. T_2 sends ν to T_1.
4. T_1 publishes $F(\mu + \nu \mod \phi, A)$ and updates A.

It is straightforward to verify that if both servers follow the protocol, then the final result $\mu + \nu \mod \phi$ is indeed the sum of the user data vectors $\mod \phi$. This result will be correct if every user's vector lies in the specified bounds for L2-norm, which implies that the sum over the integers is the same as the sum $\mod \phi$. Appropriate constraints on L will be given in the statement of theorem 3.1. Privacy of the computation protocol is summarized, together with that of the verification protocol introduced in section 3.2, in theorem 3.2.

3.2 User Data Verification Protocol

3.2.1 Overview A straightforward way of checking the L2-norm of user vector is given in [3] which works with each element and requires $O(m)$ public-key operations and ZKPs. We present a novel protocol that uses only constant or $O(\log m)$ such expensive operations thus is orders of magnitude more efficient. The key technique is that, instead of checking each elements, we check the *projections* of the user vector on some random directions. We show that some statistical properties of these projections are related to the L2-norm of the original vector. Therefore by verifying the square sum of a *small* number of such projections (in zero-knowledge), we can check if the L2-norm of a vector with a large number of elements is within a desired bound. The overhead to compute the projections is $O(m)$ but these steps only consist of arithmetic operations in the *small* field. As we will show in our experiments, the cost of such operations is very small compared to the crypto operations (It is not noticeable when $m \leq 10^5$, and is fraction of a second when m reaches 10^6).

3.2.2 Tools The verification protocol requires some standard primitives for homomorphic computation. These have appeared elsewhere, see e.g. [7, 3], and we summarize only their key properties here. All values used in these primitives lie in the multiplicative group $\mathbb{Z}_q^*$, or in the additive group of exponents for this group, where q is a 1024 or 2048-bit prime. They rely on ElGamal, RSA or discrete log functions for cryptographic protection of information.

Homomorphic commitment Given an integer value a, a homomorphic commitment to a with randomness r is written as $\mathcal{C}(a, r)$. It is homomorphic in the sense that $\mathcal{C}(a, r)\mathcal{C}(b, s) = \mathcal{C}(a + b, r + s)$. It is cryptographically hard to determine a given $\mathcal{C}(a, r)$. We say that a prover "opens" the commitment if it reveals a and r.

ZKP of knowledge A prover who knows a and r

(i.e. who knows how to open $\mathcal{A} = \mathcal{C}(a, r)$) can demonstrate that it has this knowledge to a verifier who knows only the commitment $\mathcal{A}$. The proof reveals nothing about a or r.

ZKP for equivalence Let $\mathcal{A} = \mathcal{C}(a, r)$ and $\mathcal{B} = \mathcal{C}(a, s)$ be two commitments to the same value a. A prover who knows how to open $\mathcal{A}$ and $\mathcal{B}$ can demonstrate to a verifier in zero knowledge that they commit to the same value.

ZKP for product Let $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ be commitments to a, b, c respectively, where $c = ab$. A prover who knows how to open $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ can prove in zero knowledge to a verifier who has only the commitments that the relationship $c = ab$ holds among the values they commit to.

Bit commitment Let $\mathcal{A} = \mathcal{C}(a, r)$ be a commitment to a value a where $a \in \{0, 1\}$, which is called a bit commitment. A prover who knows how to open $\mathcal{A}$ can prove in zero knowledge that it commits to either 0 or 1 (but not which).

ZKP for boundedness Let $\mathcal{A} = \mathcal{C}(a, r)$ be a commitment to a value a. Using the above methods, a prover can show that $\mathcal{A}$ contains a k-bit integer, i.e. that it encodes the same value as $\mathcal{B}_{k-1} \cdots \mathcal{B}_0$, where each $\mathcal{B}_j$ encodes 0 or 2^j. If the leading "bit" $\mathcal{B}_{k-1}$ instead encodes 0 or $L - 2^{k-1} + 1$ where $k = \lfloor \log_2 L \rfloor$, then the ZKP proves that $a \in [0, \ldots, L]$ for any k-bit positive L. Adding an additional bit which encodes 0 or $-L$ gives a proof of boundedness in the range $[-L, \ldots, L]$.

3.2.3 Protocol UDVP (User Data Verification Protocol)

Let N be a positive integer which determines the number of challenges, and sets the statistical precision of the verification. The protocol is carried out between each user and the two servers. The execution will be identical for each user so we drop the user index in the notation.

1. **Setup**: After all users send their data to all servers, T_1 broadcasts a random number r to T_2 and all users. Using a public PRG (pseudo-random generator) and r as the random seed, all players generate N independent m-dimensional challenge vector $c_k \in \{-1, 0, 1\}^m$ with each of its elements generated with IID probabilities $\{\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\}$, for $k = 1, \ldots, N$.
2. **Projection & Commitment**: For $k = 1, \ldots, N$, the user computes $x_k = c_k \cdot u \mod \phi$, $y_k = c_k \cdot v \mod \phi$, and $s_k = c_k \cdot (u + v) \mod \phi$. Let $s_k = x_k + y_k + b_k$ over the integers, then b_k is either zero or $\pm\phi$. The user computes commitments $\mathcal{X}_k$ to x_k, $\mathcal{Y}_k$ to y_k, $\mathcal{S}_k$ to s_k, $\mathcal{B}_k$ to b_k and finally a commitment $\mathcal{Z}_k$ to the squared sum $z_k = s_k^2$ (computed over the large field $\mathbb{Z}_q$). The user sends all $5N$ commitments to T_1 and T_2.
3. **Consistency Check**: T_1 and T_2 exchange these values to confirm they received identical data from the user. If they do not match, the user's data is rejected.
4. **Commitments Verification**: The user opens $\mathcal{X}_k$ for T_1, and $\mathcal{Y}_k$ for T_2, for $k = 1, \ldots, N$. Both servers confirm that the openings match their data, i.e. T_1 confirms that $\mathcal{X}_k$ is a commitment to x_k and T_2 confirms that $\mathcal{Y}_k$ is a commitment to y_k. The servers communicate the results to each other. If either opening fails or if the user failed to send a complete response to the challenge vector, this user's input is rejected.
5. **Equivalence ZKPs**: For each k, the user proves in zero knowledge to both servers that $\mathcal{S}_k$ encodes the same value as $\mathcal{X}_k \mathcal{Y}_k \mathcal{B}_k$. The user then proves in zero knowledge that $\mathcal{B}_k$ encodes 0 or $\pm\phi$. Finally the user gives a product ZKP to the servers that $\mathcal{Z}_k$ encodes the square of the value that $\mathcal{S}_k$ encodes. If any of these proofs fail, the user's input is rejected.
6. **Boundedness ZKP**: The servers computes the product $\mathcal{Z} = \prod_{k=1}^{N} \mathcal{Z}_k$. The user then provides a ZKP that $\mathcal{Z}$ encodes a value in the range $[0, NL^2/2]$. If this proof succeeds, the user's input is accepted and added to the total.

All the ZKPs used should be implemented non-interactively using the Fiat-Shamir heuristic so that the users can upload the data to the servers in a batch without further interaction.

Field/Group Sizes

The protocol assumes that the size of the cryptographic field $\mathbb{Z}_q$ used for commitments and ZKPs is much larger than the "small" group $\mathbb{Z}_\phi$ used for additive secret-sharing. A transition happens when z_k is computed from s_k. The value of s_k lies in the small group, while $z_k = s_k^2$ is computed in the large field. The sum $z = \sum_{i=1}^{n} z_k$ should be less than q to avoid modular reduction of z in the large field. This will almost surely be true. Since ϕ is typically 64 bits or less, z_k will have at most 128 bits, while z will be at most $128 + \log_2 n$ which is much less than 1024.

THEOREM 3.1. *Let $|d|_2$ denote the L2-norm of user vector d, and L be the specified bound on this norm. Define $\delta = L^2/|d|_2^2$. Then if $|d|_2 < L$, and further $\delta > 2$, the probability that a user vector is (incorrectly) rejected is at most:*

$$Pr[z > NL^2/2] \leq \left(\tfrac{\delta}{2}\exp(1-\tfrac{\delta}{2})\right)^N$$

If instead $|d|_2 > L$, the probability that a user vector is (incorrectly) accepted is at most:

$$Pr[z < NL^2/2] \leq \left(\left(\tfrac{7}{8} - \tfrac{5}{24}\delta + \tfrac{75}{288}\delta^2\right)\exp\left(\tfrac{1}{2}\delta - \tfrac{5}{12}\delta^2\right)\right)^N$$

Furthermore, these bounds are valid using modular arithmetic as per the above protocol if L satisfies:

$$L \leq \phi/\max(56.5\sqrt{m}, 2n)$$

where n is the number of users and m is the vector dimension.

The first (false rejection) bound is quite steep. A drop-off of 2^{-N} is achieved for $\delta \approx 5.3566$. If $\delta = 4$ (user vector norm is one half of L), then the roll-off is 0.736^N. The second (false acceptance) bound is considerably shallower. In the limit as $\delta \to 0$, the bound is $\frac{7}{8}^N$. For $\delta = 0.25$ (user vector norm is twice L), the bound is 0.9265^N, while for $\delta = 0.5$, it is 0.9672^N.

THEOREM 3.2. *The computation is private. Furthermore, for the purpose of verifying $z < NL^2/2$, UDVP is zero-knowledge. When used to bound $|d|_2$, only in the event of false rejection does it reveal information about valid user data other than its validity. And this leakage is exactly the fact that $z > NL^2/2$.*

The proof of theorem 3.2 consists of standard simulation of a secret sharing scheme and straightforward invocation of sequential composition ZKPs theorem [20], and is omitted. The proof of theorem 3.1 is given in appendix.

Note that what the protocol actually verifies is whether $z < NL^2/2$. For this purpose it is zero-knowledge in that the verifier learns nothing except the fact when it is true. However, this assertion is not completely equivalent to $|d|_2 < L$ and theorem 3.1 quantifies its effectiveness in bounding $|d|_2$. There is a small probability of false rejection where $|d|_2 < L$ but $z > NL^2/2$. In this case the verifier learns that at least one of the projections is large. This does not violate the zero-knowledge property of the ZKP with regard to $z < NL^2/2$ because in that context this is treated as a failed proof and there is no need to protect this information. In contrast, this is considered a leakage in our system since the user data is actually valid (thus should be protected). We believe such leakage is acceptable in most applications because (1) the false rejection probability can be reduced exponentially by using large N and (2) the leakage is very small: the verifier only learns no more than the fact that at least one of the projections is large. In particular it does not learn *which* projection is large.

COMPLEXITY The protocol computes $O(N)$ commitments and square ZKPs. In addition, the boundedness ZKP at the last step involves $O(\log(NL))$ steps. As theorem 3.1 shows, the failure probability is exponentially decreasing with N. In practice a small constant N (e.g. 50) can provide fairly good guarantee. So the total cost caused by the expensive large integer operations is $O(\log(L))$. In most cases L is either constant or polynomial in m (recall that L is the bound on the L-2 norm of an m-dimensional vector). Therefore the number of large integer operations is bounded by $O(\log m)$. This is far superior to using standard techniques which requires $O(m)$ such operations.

4 Simulations of Typical Behavior

The bounds we derived earlier show that *any* user vector whose L2-norm is substantially below L will almost surely be accepted, while any vector that is substantially above will surely be rejected. In terms of actual behavior however, the tail bounds we derived may not be very tight. Here we present some simulations for typical user data to show what behavior would be expected. Simulation could also potentially be useful to honest or dishonest users: in either case, a user with an actual input vector d_i can determine through simulation the probability of that value being accepted by the server. We choose 3 specific cases:

1. Random uniform values: every component $d[j]$ of the user vector is drawn from the same uniform distribution.
2. Zipf distribution: component $d[j]$ has value proportional to $1/j$.
3. Single element: only one value in in the user vector is non-zero.

In all cases, user vectors are normalized so their L2-norm is fixed at some value V_d. We will vary this value relative to the threshold L and determine the probability of acceptance.

The first two cases are representative of likely user data, e.g. case 1 could represent ratings for movies while case 2 could represent word counts in email or text messages. The third case is representative of a user who wants to bias the total by using a maximum value for one item. A second reason for these choices is that cases 1 and 3 represent probable extremes of distributions of user vectors. All sums s_k are sums of 3-valued $s_k[j]$. The more terms in this sum and the more similar those terms, the closer will be the final distribution to a gaussian. The s_k produced by case 1 are almost perfectly gaussian. The s_k for Zipf distributed data are mixtures of terms with very different weights, and

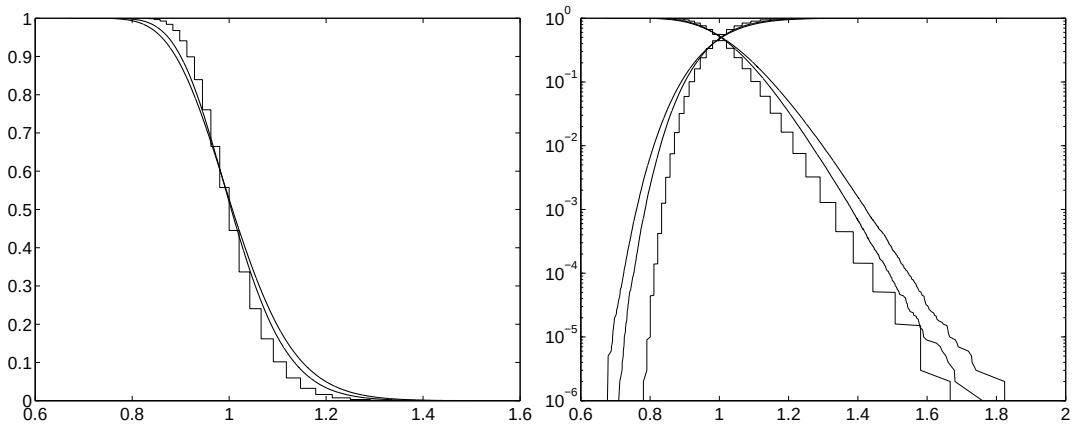

Figure 1: (a) Linear and (b) log plots of probability of user input acceptance as a function of V_d/L for $N = 50$. (b) also includes probability of rejection. In each case, the steepest (jagged curve) is the single-value vector (case 3), the middle curve is Zipf vector (case 2) and the shallow curve is uniform vector (case 1)

are "less" gaussian. Finally, the s_k for single-element vectors retain a 3-valued distribution and are very far from gaussian. Any distribution the user can produce will be a sum of such $s_k[j]$, and will probably lie between the extremes of cases 1 and 3.

The simulations used $N = 50$, $m = 100$, and were repeated 10^6 times. Figure (1) shows probabilities of acceptance or rejection for the 3 cases as a function of the ratio V_d/L. Increasing N by a factor α should cause the log plots to scale by α in their y-values. When $N = 50$, the upper tail bound from theorem 1 has an asymptotic slope of 25 in $\log(Pr)$ vs. $\log(\delta)$ plots. The lower tail bound slope is significantly shallower because of "saturation" of the probability to $\frac{7}{8}^N$as $\delta \to 0$. The x-axes in figure 1 involve $|d|/L$ which is $1/\sqrt{\delta}$. The expected slopes from the tail bounds would be 12.5 for the rejection probability curve, and less for the acceptance curve. The actual slope observed for rejection is about 50, while it is around 35 for acceptance. So the typical threshold behavior for the probabilistic L2-bound is much sharper than the asymptotic bounds from theorem 3.1.

5 Implementation and Evaluation

We have implemented the protocols in Java using a NativeBigInteger implementation from the I2P anonymous network (http://www.i2p.net/). We measured the performance on a 2.8GHz Xeon. All tests were carried out using El-Gamal commitments and ZKPs [7] in a large field $\mathbb{Z}_q$ for a 1024-bit prime q. N was 50, and L was either a 40-, 20- or 10-bit number. The basic bit commitment ZKP takes 33.7 ms for the verifier and 57.3 ms for the prover. The square ZKP takes 35.7 ms verifier time and 24.3 ms prover time. Figure 2 plots prover (user) and verifier times for the L2-norm validation protocol as a function of the vector size m. Both were dominated by cryptographic operations in these experiments, even at $m = 10^6$. Other steps, such as random vector generation, or computation of all the products $c_k \cdot d$ by the prover, took a fraction of a second. At $m = 10^6$, verifying one user's vector takes only a few seconds. In contrast, using standard techniques which requires $O(m)$ square ZKPs, as is done in [3], both the verifier and prover time is in hours (about 10 hours for the verifier and 6.7 hours for the prover). Our protocol is orders of magnitudes more efficient. The server can easily process over 15000 users each day with a *single* PC. Since the protocol for each user is independent of each other, the server can support larger user group with a cluster, which is a common practice for today's service providers. Most of the applications such as collaborative filtering require only infrequent update (once per day or more), the performance we obtained is more than enough for them. The communication overhead is also very small since it passes very few large integers. The communication per client is only a few kilobytes, while other solutions require some hundreds of megabytes.

6 Conclusion and Future Work

This paper shows that private vector addition with verification can be done at extremely low cost in many realistic settings. It opens the door to a variety of applications built on such primitive. Most components described in this paper have been implemented and the source code is made available as a toolkit for the general public. Please visit

http://www.cs.berkeley.edu/∼duan/research/p4p.html for download and other information. In the near future, we plan to add some "middle tier" common statistical aggregate components such as ANOVA, SVD, etc. Supporting general algorithms within the paradigm (VSS over small field and random projection-based verification) is also being actively explored. Our goal is to make it a useful tool for developers in data mining and others to build privacy preserving real-world applications.

References

[1] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *SIGMOD '00*, pages 439–450. ACM Press, 2000.

[2] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: the SuLQ framework. In *PODS '05*, pages 128–138, New York, NY, USA, 2005. ACM Press.

[3] J. Canny. Collaborative filtering with privacy. In *IEEE Symposium on Security and Privacy*, pages 45–57, Oakland, CA, May 2002.

[4] J. Canny. Collaborative filtering with privacy via factor analysis. In *SIGIR '02*, pages 238–245. ACM Press, 2002.

[5] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *IEEE Foundations of Computer Science*, pages 383–395, 1985.

[6] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun. Map-reduce for machine learning on multicore. In *NIPS 2006*, 2006.

[7] R. Cramer and I. Damgård. Zero-knowledge proof for finite field arithmetic, or: Can zero-knowledge be for free? In *CRYPTO '98*. Springer-Verlag, 1998.

[8] R. Cramer, I. Damgård, and J. B. Nielsen. Multiparty computation from threshold homomorphic encryption. In *EUROCRYPT '01*, pages 280–299. Springer-Verlag, 2001.

[9] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In *WWW '07*, pages 271–280, New York, NY, USA, 2007. ACM Press.

[10] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *PODS '03*, pages 202–210, New York, NY, USA, 2003. ACM Press.

[11] W. Du, Y. Han, and S. Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *SIAM International Conference on Data Mining*, pages 222–233, 2004.

[12] W. Du and Z. Zhan. Using randomized response techniques for privacy-preserving data mining. In *KDD '03*, pages 505–510. ACM Press, 2003.

[13] Y. Duan and J. Canny. Zero-knowledge test of vector equivalence and granulation of user data with privacy. In *IEEE GrC 2006*, 2006.

[14] Y. Duan, J. Wang, M. Kam, and J. Canny. A secure online algorithm for link analysis on weighted graph. In *Proceedings of the Workshop on Link Analysis, Counterterrorism and Security, SIAM Data Mining Conference, 2005*, pages 71–81.

[15] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC 2006*, volume 3876 of *LNCS*, pages 265–284. Springer, 2006.

[16] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *PODS '03*, pages 211–222. ACM Press, 2003.

[17] M. Fitzi, M. Hirt, and U. Maurer. General adversaries in unconditional multi-party computation. In *ASIACRYPT 99*, volume 1716 of *LNCS*, pages 232–246. Springer-Verlag, 1999.

[18] R. Gennaro, M. O. Rabin, and T. Rabin. Simplified vss and fast-track multiparty computations with applications to threshold cryptography. In *PODC '98*, pages 101–111. ACM Press, 1998.

[19] O. Goldreich. *Foundations of Cryptography: Volume 2 Basic Applications.* Cambridge University Press, 2004.

[20] O. Goldreich and Y. Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, 1994.

[21] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989.

[22] M. Hirt and U. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In *PODC '97*, pages 25–34, 1997.

[23] M. Hirt and U. Maurer. Player simulation and general adversary structures in perfect multiparty computation. *Journal of Cryptology*, 13(1):31–60, 2000.

[24] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, 22(1):89–115, 2004.

[25] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1026–1037, 2004.

[26] H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. On the privacy preserving properties of random data perturbation techniques. In *ICDM '03*, page 99, Washington, DC, USA, 2003. IEEE Computer Society.

[27] M. Kearns. Efficient noise-tolerant learning from statistical queries. In *STOC '93*, pages 392–401, New York, NY, USA, 1993. ACM Press.

[28] Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of cryptology*, 15(3):177–206, 2002.

[29] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[30] T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO '91*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer-Verlag, 1991.

[31] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Application of dimensionality reduction in recom-

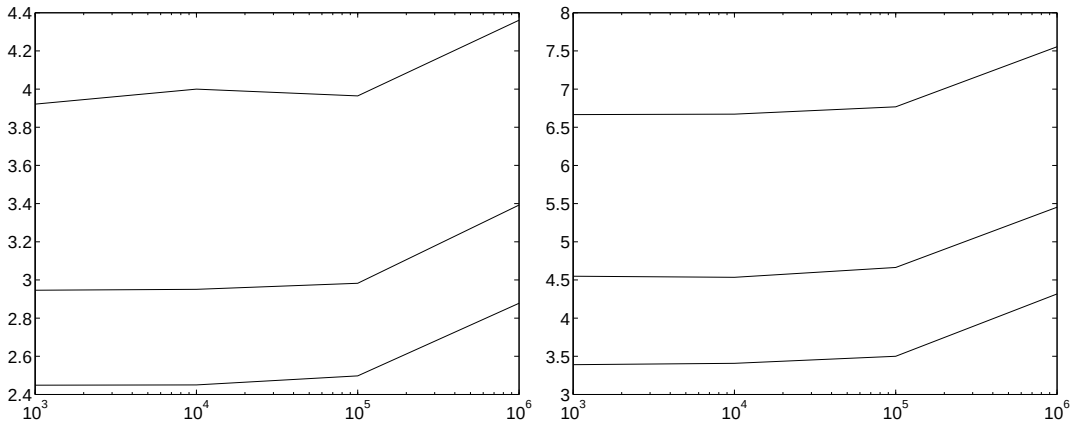

Figure 2: (a) Verifier and (b) prover times in seconds for the validation protocol with $N = 50$, where (from top to bottom) L has 40, 20, or 10 bits. The x-axis is the vector length m.

mender system – a case study. In *ACM WebKDD 2000 Web Mining for E-Commerce Workshop*, 2000.

[32] G. W. Stewart and J.-G. Sun. *Matrix Perturbation Theory*. Academic Press, 1990.

[33] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *KDD '02*, pages 639–644, New York, NY, USA, 2002. ACM Press.

[34] J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *KDD '03*, pages 206–215, New York, NY, USA, 2003. ACM Press.

[35] R. Wright and Z. Yang. Privacy-preserving bayesian network structure computation on distributed heterogeneous data. In *KDD '04*, pages 713–718, New York, NY, USA, 2004. ACM Press.

[36] Z. Yang, S. Zhong, and R. N. Wright. Privacy-preserving classification of customer data without loss of accuracy. In *SDM 2005*, 2005.

A Proof of Theorem 1

We first present proofs for the tail bounds assuming the total s_k on each round exactly represents the weighted sum of user vectors $s_k = \sum_{j=1}^{m} c_k \cdot d$. Because the sums are actually computed $\mod \phi$, s_k may differ by a multiple of ϕ from the total over the integers. We deal with modular arithmetic effects later in this section.

Statement Let $z_k = s_k^2$, and $z = \sum_{k=1}^{N} z_k$. Let $V = E[z_k]$ for $k = 1, \ldots, N$ and $\delta = (L/|d|_2)^2 = L^2/(2V)$. Then the probability that a user input fails the test is probability that $Pr[z > NL^2/2] = Pr[z > \delta NV]$, and we claim that

$$Pr[z > \delta NV] \leq \left(\tfrac{\delta}{2}\exp(1-\tfrac{\delta}{2})\right)^N \qquad \text{where } \delta > 2$$

Conversely, with the same definitions and if $\delta < 1$, the user will pass the test if $Pr[z < NL^2/2] = Pr[z < \delta NV]$, which has a bound

$$Pr[z < \delta NV] \leq \left(\left(\tfrac{7}{8} - \tfrac{5}{24}\delta + \tfrac{75}{288}\delta^2\right)\exp\left(\tfrac{1}{2}\delta - \tfrac{5}{12}\delta^2\right)\right)^N$$

where $0 < \delta < 1$

Proof Since s_k is a sum of independent random variables, the pdf of s_k typically has gaussian-decay tails, but the squares $z_k = s_k^2$ in general are not gaussian and have simple exponential tails. We can still prove Chernoff-style bounds for the tails of z that show exponential decrease in the number of trials, but the bounds have only simple-exponential decrease away from the mean. In all cases, we use bounds of the moments of z_k which are derived later in Lemma 1.

Upper Tail

First, for the upper tail let $\delta > 1$, and since $E[z] = NV$, we evaluate

$$Pr[z > \delta NV] = Pr[\exp(tz) > \exp(t\delta NV)]$$

and applying a Markov bound we obtain

$$Pr[z > \delta NV] \leq \frac{E[\exp(tz)]}{\exp(\delta NV)} \tag{1.1}$$

and since the z_k are independent for $k = 1, \ldots, N$, we can factor the expected value as the product of $E[\exp(tz_k)]$. Since we have all the moments of z_k, we

can compute this value as a power series:

$$\begin{aligned} E[\exp(tz_k)] &= \sum_{i=0}^{\infty} t^i E[z_k^i]/(i!) \\ &\leq \sum_{i=0}^{\infty} (tV/2)^i (2i)!/(i!)^2 \\ &= \sum_{i=0}^{\infty} (tV/2)^i \binom{2i}{i} \end{aligned}$$

and since $\binom{2i}{i} \leq 4^i$, this series will converge so long as $2tV < 1$. The series is then geometric, and has a bound of:

$$E[\exp(tz_k)] \leq \frac{1}{1-2tV}$$

and substituting into (1.1) gives

$$Pr[z > \delta NV] \leq \frac{1}{(1-2tV)^N \exp(t\delta NV)} \tag{1.2}$$

and this bound is optimized by maximizing $(1-2tV)\exp(t\delta V)$. Taking derivatives and solving gives $t = 1/(2V) - 1/(V\delta)$. The bound is valid so long as $0 < t < 1/(2V)$, which is true if $\delta > 2$. Substituting, we obtain:

(1.3)

$$Pr[z > \delta NV] \leq \left(\tfrac{\delta}{2}\exp(1-\tfrac{\delta}{2})\right)^N \qquad \text{where } \delta > 2$$

Lower Tail

Now let $\delta > 0$, $E[z] = NV$, we evaluate

$$Pr[z < \delta NV] = Pr[\exp(-tz) > \exp(-t\delta NV)]$$

for $t > 0$, and applying a Markov bound we obtain

$$Pr[z < \delta NV] \leq \frac{E[\exp(-tz)]}{\exp(-t\delta NV)} \tag{1.4}$$

the expected value factors as before into terms $E[\exp(-tz_k)]$. The expansion is an alternating sum which is difficult to bound, so instead we truncate it using the inequality

$$\exp(-y) \leq 1 - y + y^2/2$$

which holds for all $y > 0$. This gives the bound:

$$\begin{aligned} E[\exp(-tz_k)] &\leq E[1 - tz_k + t^2 z_k^2/2] \\ &= 1 - tV + \frac{t^2}{2} E[z_k^2] \\ &\leq 1 - tV + \frac{3}{2} t^2 V^2 \end{aligned} \tag{1.5}$$

where the last step used the moment bounds from Lemma 1. Substituting into (1.4) gives

$$Pr[z < \delta NV] \leq \left((1 - tV + \tfrac{3}{2}t^2V^2)\exp(tV\delta)\right)^N \tag{1.6}$$

Minimizing the RHS involves solving a quadratic equation which is a function of δ. The solution can be approximated as $t \approx (1/2 - 5/12\delta)/V$. We can use this value as a bound in any case, giving:

(1.7)

$$Pr[z < \delta NV] \leq \left((\tfrac{7}{8} - \tfrac{5}{24}\delta + \tfrac{75}{288}\delta^2)\exp(\tfrac{1}{2}\delta - \tfrac{5}{12}\delta^2)\right)^N$$

Dealing with Modular Arithmetic

In order for the secret shares not to leak information about user data, modular arithmetic is used. We use the notation $x[i]$ for the i^{th} component of the vector x. We denote by $\bar{s}_k = \sum_{j=1}^{m} c_k[j]d[j]$ the sum over the integers, and by s_k this sum reduced $\mod \phi$, which is what the protocol actually computes. Then we have

$$\bar{s}_k = s_k + w\phi$$

for some integer w. The modular arithmetic provides additional ways for the user to cheat. e.g. the user might set some components of her vector to $\phi/2$. If an even number of those are included in the checksum, they will be removed by the modular arithmetic, leading to a small s_k. However, we show now that any such "large" components will cause the protocol to fail almost surely. We consider the following two cases:

1. All components $d[i]$ of the user's vector are in the range $[-4L, 4L]$
2. Some component $d[i]$ has magnitude larger than $4L$.

Note that case 1 includes both legal and illegal inputs, since the largest legal magnitude for any component is L. Case 2 vectors have overall magnitude greater than L and are strictly illegal.

Case 1: If all components of the user vector are in the range $[-4L, 4L]$, then the maximum variance of this vector is $V = 32mL^2$. The reduced s_k will be equal to $\bar{s}_k$ as long as $\bar{s}_k$ is in the range $\mathbb{Z}_\phi$, i.e. as long as $|\bar{s}_k| \leq \phi/2$. By setting δ to the ratio of squared limit over variance $\delta \geq \phi^2/(32mL^2)$, and $N = 1$ we can use the upper tail bounds computed earlier to bound a single s_k.

$$Pr[|\bar{s}_k| > \phi/2] = Pr[z_k > \delta V] \leq \left(\tfrac{\delta}{2}\exp(1-\tfrac{\delta}{2})\right)$$

A typical safe value would be $\delta = 100$, giving a failure probability of 2.6×10^{-20}. The bound L must satisfy $L \leq \phi/\sqrt{32\delta m}$, which for $\delta = 100$ becomes $L \leq \phi/(56.5\sqrt{m})$. This constraint would normally be satisfied in any practical system, because L must be small enough to allow $\bar{s}_k$ totals to be computed without wrapping $\mod \phi$. That is if there are n users, the bound L should be such that $nL \leq \phi/2$, because a legal

user input may have a value of L in one element only. Satisfying both constraints gives us the result: $L \leq \min(\phi/(56.5\sqrt{m}), \phi/(2n))$ or $L \leq \phi/\max(56.5\sqrt{m}, 2n)$

Case 2: Some $|d[i]| > 4L$. Fix this i, and let $\bar{s}_{-i}$ denote the sum $\sum c[j]d[j]$ of all terms $j \neq i$ over the integers. Now either $\bar{s}_{-i}$ is in some range $[-2L, 2L] + k\phi$ or it isnt (we say it is "legal" it is is in such a range). The final total $\bar{s} = c[i]d[i] + \bar{s}_{-i}$ differs from $\bar{s}_{-i}$ by either 0 or $d[i]$ where $4L \leq |d[i]| \leq \phi/2$. If $\bar{s}_{-i}$ is legal, then $\bar{s}_{-i} \pm d[i]$ must be illegal, which has probability 1/2. If $\bar{s}_{-i}$ is illegal to begin with, then at most both the offsets $\pm d[i]$ will be legal, which again has probability 1/2. If p is the probability that $\bar{s}_{-i}$ is legal at first, the probability that $\bar{s}$ is legal is at most $\frac{1}{2}p + \frac{1}{2}(1-p) = \frac{1}{2}$.

Now let $q \leq N$ be the number of challenges for which $\bar{s}_k$ is illegal, i.e. the number of k for which $\bar{s}_k > 2L$. For each of these $z_k > 4L^2$ and the total z will be at least $4qL^2$. The overall user data verification will (incorrectly) succeed if $z < NL^2/2$, which can only happen if $q < N/8$. The probability that this happens is the tail of a Bernoulli distribution over uniform trials with probability $\geq \frac{1}{2}$. Using standard formulae [29], this probability is bounded by:

$$Pr[z < NL^2/2] \leq 0.8173^N$$

This probability is strictly less than the lower tail bound derived above which is never better than $\frac{7}{8}^N = 0.875^N$. So the latter bound dominates, and we do not separately quote the probability for modular wrap-around error.

A.1 Lemma 1 For independent random variables $c[j]$ in $\{-1, 0, 1\}$ with probabilities $\{\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\}$ respectively, and let $s = \sum_{j=1}^{m} c[j]d[j]$, and $z = s^2$. Then all positive moments of z satisfy:

$$E[z^q] \leq \frac{(2q)!}{q!2^q} V^q = (1 \cdot 3 \cdot 5 \cdots (2q-1))V^q \leq (qV)^q$$

where $V = E[z] = E[s^2]$ as before.

Proof We rewrite the sum for each moment as: $E[z^q] = E[s^{2q}] = E[(\sum_{i=1}^{m} s[i])^{2q}]$ and fully expanding the last term gives:

(1.8)
$$E[z^q] = \sum_{r=1}^{2q} \sum_{i_1+\cdots+i_r=2q}^{1\leq i_1,\ldots,i_r\leq 2q} \binom{2q}{i_1, \ldots, i_r} E[s[j_1]^{i_1} \cdots s[j_r]^{i_r}]$$

where $1 \leq j_1 < j_2 < \cdots < j_r \leq m$. Next we notice that each $s[j]$ is symmetric: $Pr[s[j] = v] = Pr[s[j] = -v]$. So every term containing an odd power of some $s[j]$ has expected value zero. wlog we can assume that every index $i_1, \ldots, i_r$ in the expression above is even.

The expected values in the last formula can be computed directly since the s_j are independent:

$$E[s[j_1]^{2i_1} \cdots s[j_r]^{2i_r}] = \frac{1}{2^r} d[j_1]^{2i_1} \cdots d[j_r]^{2i_r}$$

Rewriting (1.8) using this expansion, and using only even powers gives:

(1.9) $$E[z^q] = \sum_{r=1}^{2q} \sum_{i_1+\cdots+i_r=q}^{1\leq i_1,\ldots,i_r\leq q} \binom{2q}{2i_1, 2i_2, \ldots, 2i_r} 2^{(-r)} d[j_1]^{2i_1} \cdots d[j_r]^{2i_r}$$

In order to simplify this last expression, we consider the expansion of $(2V)^q$ which is:

(1.10) $$(d[1]^2 + \cdots + d[m]^2)^q = \sum_{r=1}^{q} \sum_{i_1+\cdots+i_r=q}^{1\leq i_1,\ldots,i_r\leq q} \binom{q}{i_1, i_2, \ldots, i_r} d[j_1]^{2i_1} \cdots d[j_r]^{2i_r}$$

which contains exactly the same products of $d[j]$'s. We take the ratio of the coefficients of $d[j_1]^{2i_1} \cdots d[j_r]^{2i_r}$ in (1.9) and (1.10), giving

(1.11) $$R = 2^{-r} \binom{2q}{2i_1, \ldots, 2i_r} / \binom{q}{i_1, \ldots, i_r}$$

We expand this first as

$$\frac{(2q)!}{q!} \frac{i_1!}{(2i_1)!} \cdots \frac{i_1!}{(2i_r)!} 2^{-r}$$

and notice that $\frac{i_j!}{(2i_j)!} \leq 2^{-i_j}/(i_j)!$. Making these substitutions gives

$$R \leq \frac{(2q)!}{q!} \frac{1}{2^{i_1} i_1!} \cdots \frac{1}{2^{i_r} i_r!} 2^{-r} = (q)^q \frac{1}{i_1! \cdots i_r! 2^r}$$

and finally it is easy to show that $i_1! \cdots i_r! 2^r \geq 2^q$. This can be done inductively by starting with $r = q$, and all $i_j = 1$, and "walking" to any desired partition $i_1 + \cdots + i_r$ of q, each step merging some i_j which is $= 1$ with another. The number of groups r decreases by 1 at each step which reduces 2^r by 2, but some i_l is incremented at the same time, and so $i_l!$ is multiplied by *at least* 2. Substituting for these expressions in the denominator of R in the last equation gives:

$$R \leq \frac{(2q)!}{q!} 4^{-q}$$

Now if we multiply equation (1.10) by this value, we guarantee that every term in its expansion is at least as great as the coefficient in equation (1.9). Or in other words,

$$E[z^q] \leq \frac{(2q)!}{q!2^q} V^q = (1 \cdot 3 \cdot 5 \cdots (2q-1))V^q \leq (qV)^q \qquad \text{QED}$$