Practical Private Computation of Vector Addition-Based Functions or: Can Privacy be for Free?

John F. Canny
Yitao Duan

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2006-12
http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-12.html

February 8, 2006

Acknowledgement

This work was supported by National Science Foundation award #EIA-0122599 (Title: ``ITR/SI:Societal Scale Information Systems: Technologies, Design, and Applications'').

Practical Private Computation of Vector Addition-Based Functions or: Can Privacy be for Free? *

John Canny and Yitao Duan

Computer Science Division
University of California, Berkeley
Berkeley, CA 94720, USA
{jfc, duan}@cs.berkeley.edu

Abstract. In this paper we explore private computation built on vector addition. Vector addition is a surprisingly general tool for computing private aggregates over many users' data. Examples including linear algorithms like voting and summation, but also many non-linear algorithms like SVD, regression, ANOVA and several machine learning algorithms based on Expectation Maximization (EM). These methods aggregate user data only in certain steps, such as conjugate gradient, which are *linear* in per-user data. In this paper we introduce a new and highly-efficient private vector addition protocol which is a blend of P2P and client-server. Secret-shared arithmetic operations are done *over small fields* (e.g. 32 or 64 bits), so that private arithmetic operations have the same cost as normal arithmetic. Verification of user data is required, which uses large-field public-key arithmetic (1024 bits or more) and homomorphic computation. The main result of this paper is a random projection method for verification that requries only a *logarithmic* number of large-field operations. Overall, the protocol is dominated by the (linear) time to do small-field operations on user data. Essentially our solution provides user privacy for free from a server or client's perspective. In experiments, addition and verification of a million-element vector takes approximately three seconds of server time and about five seconds of client time on state-of-the-art PCs.

1 Introduction

Many kinds of analysis depend on data from a group of users. Examples include voting, summation, SVD, regression, and ANOVA. The standard algorithms for these functions use gradient steps which sum vector data from the users. Machine learning algorithms based on the EM (Expectation Maximization) method are

* This work was supported by National Science Foundation award #EIA-0122599 (Title: "ITR/SI: Societal Scale Information Systems: Technologies, Design, and Applications").

another broad class that rely on vector-addition steps. E-commerce recommendations are based on detailed purchase histories of individual users. Location-based services are a new and fast-growing industry, and they perform best when user location histories are available. Many hospitals are moving fast to electronic medical records (EMR), which opens up new possibilities for large-scale studies of treatments, drug interactions or lifestyle influences on health. Finally, many new intranet productivity tools attempt to mine user expertise and interest from the contents of users' desktops, email and web use.

In all these cases, it is very important to protect user privacy to the maximum extent possible. At the same time, it is important to provide the same service with no loss of accuracy (which can mean lost revenue for the provider). 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.

For the tasks considered here, private arithmetic has come closest to providing a practical solution [1–3]. While both addition and multiplication are possible in most of these schemes, the practical overhead for multiply is much higher. As shown in [3], non-trivial and non-linear computation (in that case a Singular-Value Decomposition) can be done using an iterative algorithm with vector-addition aggregation steps. Other examples of addition-only calculations include EM-based collaborative filtering [4], the GaP algorithm for discrete data mining [5]), link analysis algorithms such as PageRank [6], HITS [7], and a new online HITS algorithm [8]. 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.

On a typical computer today there is a six order of magnitude difference between the crypto exponentiations needed for homomorphic computation (order of milliseconds) and regular arithmetic operations used in secret-sharing (fraction of a nano-second). Both homomorphic arithmetic [3] and VSS (Verifiable Secret Sharing) [9] rely on public-key operations for verification. Even fast-track VSS [2] does not reduce the asymptotic number of crypto operations. In this paper we present a vector addition protocol that uses a linear number of *small-field* operations, and a *logarithmic* number of crypto operations in the size of the user data. Thus, private computation is essentially "free" even counting constant factors. It is orders of magnitude faster than other approaches for moderate or large datasets. We give experimental results which show that it is extremely fast for typical data (order of seconds for a million-element vector). We argue that this moves private computation into the realm of the practical for many Internet and intranet applications. The major technical contributions of our work can be summarized as follows:

– We propose a realistic, hybrid architecture between peer-to-peer (P2P) and the client-server model. This architecture exploits the heterogeneity of the players and greatly simplifies multi-party computation.

- A very efficient probabilistic data validation protocol that ensures, with high probability, a malicious user cannot exert large amount of influence on the computation.
- A scalable probabilistic protocol for verifying the computation. This protocol requires only $O(\log m)$ large-field operations, and $O(m)$ small-field operations, where m is the size of user vector.

2 P4P Basics

Our approach is called Peers for Privacy (P4P) for its unique architecture. The protocol assumes a single computer called the *server*, which is operated by a service provider. We also assume a (small) number of designated *privacy peers* (PP) who participate in the computation, and support the VSS. In contrast to previous work, privacy peers are assumed to belong to users in the community. They are not required to be honest, and the protocol ensures that they cannot break the privacy of the protocol without the server's help. In practical P2P systems such as Gnutella and Napster, a small fraction of the users in the community provide most of the storage to the others. The existence of such altruistic users is a pervasive phenomenon in communities. For workplace privacy, the peer would be a special employee (e.g. a union representative). For e-commerce privacy, a group of users may choose a well-respected peer. Or the service may be provided for a fee by a third-party commercial "trust provider". In certain cases, it may be feasible to distribute the VSS among service providers. For example, two hospitals may wish to mine data from hospital records shared among them. However, we will argue later that the incentives for collusion among servers and privacy peers are very different, and that in practice the protection offered by this arrangement is very good compared to multi-server approaches. And economically, the P4P approach is superior since it requires no additional resources on the server side.

Since the privacy peer is not trusted and cannot compromise user data, there are many options for it. While the "large institutional server" model is more familiar, the "small individual peer" model has many possibilities. In the latter model, we assume a much smaller ratio between peer and users. e.g. while a single server would normally service millions of users, an individual privacy peer would support a few hundreds to perhaps tens of thousands of users. The asymmetry between server and privacy peer is an important source of security because they have different incentives. A service provider will benefit from accessing sensitive data from many users, but much less so from data from a few users. A privacy peer on the other hand, may be interested in particular data about a few individuals, but has little to gain from information from a large number, except perhaps to sell it to the main service provider (it would be useless to anyone else). But to be able to gain access to a significant fraction of its users' data, a service provider would have to corrupt many privacy peers.

A final disincentive to a server to cheat is the high risk of discovery in P4P. Compromising privacy requires server and a peer to exchange data, and both

will be aware of the cheating. In a large P4P system, a server would cooperate with many peers, each peer serving a fraction of the community. For a server to get data from a significant number of the clients, it would have to conspire with many privacy peers. Any of these peers may chose to expose the server's cheating.

The protocols we describe are 2-way multi-party computations, or compositions of those. While 2-way MPC might seem intuitively less secure than a k-way MPC for larger k, such intuition is meaningless without knowledge of the dependencies between the parties. A k-way MPC among parties with similar incentives is not necessarily more secure than a 2-way MPC among dissimilar parties. Service providers in the same business who would be the best candidates for k-way MPC unfortunately share the same incentives and so their behavior is highly dependent. In the following we describe our protocols in a 2-way setting, carried out between a server and a privacy peer who are referred to as *talliers*.

2.1 Assumptions

We assume all n users have access to secure channels with the server and the privacy peer(s). Let a_i be private user data for user i and A be public information. Both can be matrices of arbitrary dimensions. The most general form of computation we support can be expressed as $A' = F(\sum_{i=1}^{n} d_i, A)$, where $d_i = G(a_i, A)$ is an m-dimensional data vector for user i computed locally, and A' is a successor iterate to A. Both functions F and G are in general non-linear. Most gradient-based and EM algorithms can be expressed in this form. If d_i are computed locally by each user, our protocol allows calculation of the sum, and thence the next iterate A' without disclosing any information about d_i or a_i.

Let ϕ be a small integer (e.g. 32 or 64 bits). Our goal is to do limited-range integer (or fixed-point) vector addition. To provide information-hiding via randomization, we embed this integer range in the additive group of integers modulo ϕ. Since our 2-way VSS protocols involve 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.[1] Since we need signed values, we consider the specific coset representatives of the integers $\mod \phi$ 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.

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 the expected total 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 $[-L, L]$.

[1] It is not difficult to generalize our protocols to (p, q)-threshold secret sharing which requires multiplication and multiplicative inverses. In that case, ϕ should be a prime so that $\mathbb{Z}_\phi$ is a field.

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 1). Security and privacy goals follow those introduced in [3]. Namely:

1. No participants, except user i herself, should gain any information about d_i, except that:
2. User data is almost surely valid, meaning that, with high probability, $|d_i|_2 < L$ where $|d_i|_2$ denote the L2-norm of the vector d_i and L is the predefined bound. Without this step, users could submit values up to $\lfloor \phi/2 \rfloor$ and completely corrupt the computation.
3. The computation should produce the correct sum if the server and privacy peer are honest.

ADVERSARY MODELS We consider two adversary models. In both models, an adversary may actively corrupt any number of users, causing them to deviate arbitrarily from the specified protocol. The titles of the models refer to adversary's effect on the tallier.

Passive Adversary In this model, the adversary corrupts any number of users, and *passively* corrupts one tallier. That is, the adversary can read data from the tallier's memory, but the tallier continues to follow the protocol.

Active Adversary In this model, the adversary actively corrupts users and one tallier. This tallier may behave arbitrarily.

The first model is similar to the adversary model in [10]. We avoid the impossibility results of [10] by considering addition only computation.

These models are intended to model realistic scenarios for malfeasance by the talliers. The passive adversary models privacy risks due to access to data files from the server or privacy peer. The active adversary models corruption of the server or privacy peer software.

3 Related Work

Secure multiparty computation (MPC) problem dates back to Yao [11] and Goldreich et al. [12]. Important works include [13–16] etc. They provide general solutions for computing any n-ary function among n players while protecting each player's private data. Although theoretically powerful, these protocols are not practical for large scale systems, even for addition only computation. They make heavy use of public-key cryptosystems or one-way functions, applying verifications or ZKPs at most steps, and usually operate at bit-level (rather than on arithmetic values).

In [3], Canny presented a privacy preserving protocol for collaborative filtering based on SVD that is actually feasible to be implemented in a system with

moderate scale. It includes an efficient zero-knowledge proof of user data validity that restricts the amount of influence a malicious user can exert on the computation. The techniques (for both data validation and computation), based on threshold homomorphic encryptions, are applicable to any computation based on vector addition and have been used in several other applications including the IR algorithm in [4] and the link analysis algorithm in [8].

Besides homomorphic threshold encryption, VSS is also a widely used primitive in private arithmetic computation (e.g. [2]). VSS makes use of information-theoretic protection using secret-sharing, and works for fields of any size. In particular it works for fields of "normal" size (32 or 64 bits). Homomorphic computation relies on cryptographic protection, and needs large fields of 1024 bits or more.

Privacy-preserving data mining is one of the major target applications of our protocol. Existing solutions use either randomization (e.g. [17, 18]) or cryptographic techniques (e.g. [19–22]) to protect privacy. Besides sacrificing accuracy, randomization has been shown to provide very little privacy protection in many cases [23]. Our approach is in the 2nd category and provides cryptographically strong privacy. We introduce an efficient VSS computation paradigm that is based on realistic trust assumptions and allows for efficient user data validation which is generally lacking from all these works. Our protocol can thus be used as a building block for many practical data mining tasks.

4 Practical Vector Addition-Based VSS

We consider first a passive adversary which may read one tallier's data, but for which both talliers follow the protocol.

4.1 Passive Adversary

Let T_1 denote the server and T_2 one of the privacy peers. Assume $Q = \{1, \ldots, n\}$ is the initial set of qualified users. The basic computation is carried out as follows:

1. User i 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 talliers that her input is valid using the protocol that will be described in Section 4.2. If she fails to do so, both talliers exclude her from Q.
3. If enough (e.g. more than 80% of all users) 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, and T_1 sends μ to T_2.
4. T_1 publishes $A' = F(\mu + \nu \mod \phi, A)$ and updates A.

It is straightforward to see that if both talliers follow the protocol, then the final result $(\mu + \nu)$ is indeed the sum of the user data vectors $d \mod \phi$. 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 1.

In terms of protecting privacy, a simple simulation could generate random user vectors u_i and v_i which are indistinguishable from the actual data that the talliers receive. The final sum is public information, so each tallier gains no other information, except what they may learn from the verification protocol.

4.2 User Verification Protocol for Passive Adversaries

The verification protocol requires some standard primitives for homomorphic computation. These have appeared elsewhere, see e.g. [1], [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 number. They rely on El-Gamal, 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 $\mathcal{C}(a, r)$. 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 they reveal 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 $[-L, \ldots, L]$.

The Protocol

Let L be the desired bound on the L2-norm of all user vectors. Let N be an 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 talliers. The execution will be identical for each user so we drop the user index in the notation.

1. After the user sends her data to all talliers, T_1 and T_2 chose random values r_1 and r_2 respectively and exchange commitments to them. Then they exchange r_1 and r_2, verify the commitments, and compute $r = r_1 + r_2$ as the random seed for the challenge vectors c_k, for $k = 1, \ldots, N$. Each m-dimensional challenge vector is generated by hashing r, giving $c_k \in \{-1, 0, 1\}^m$, with IID probabilities $\{\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\}$. The server sends the c_k to all users.[2]

2. Each user computes $x_k = c_k \cdot u \mod \phi$, $y_k = c_k \cdot v \mod \phi$, and $s_k = c_k \cdot (u+v) \mod \phi$ for $k = 1, \ldots, N$. 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 $\mathbb{Z}_q$). These commitments are computed for every $k = 1, \ldots, N$. The user sends all $5N$ commitments to T_1 and T_2.

3. 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. The user opens $\mathcal{X}_k$ for T_1, and $\mathcal{Y}_k$ for T_2, for $k = 1, \ldots, N$. Both talliers 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 talliers 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. For each k, the user proves in zero knowledge (to the server) 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 server 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. The server 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.

Field 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" field $\mathbb{Z}_\phi$ used for secret-sharing. A transition happens when z_k is computed from s_k. The value of s_k lies in the small field, while $z_k = s_k^2$ is computed in the large field. The sum $z = \sum_{i=1}^{n} z_k$

[2] If the hash function used to compute c_k from r is public information, r can be sent directly to users to avoid the communication cost of sending the vectors c_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 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.

4.3 Active Adversary

To deal with some corrupted talliers we must increase the number of talliers and rely on consensus. Typically this is done with threshold secret sharing. General threshold secret sharing would require byzantine agreement between talliers, and a more complex verification protocol. This can in fact be done in the framework we have described with similar efficiency: By checking computation on *projections* of the user data shares using the challenges vectors rather than the original vectors themselves. But it would lead to a quite complicated protocol which would be challenging to implement.

However, there is a simpler approach which is much more practical to implement, and which matches better with realistic threat models. We make the following assumptions:

1. The privacy peer is much more likely to be corrupted than the server. Since it is a user-owned machine, it runs a high risk of virus corruption or even corruption by its owner.
2. The server rarely has incentives to distort the tallying, since they typically benefit from the most accurate tallys. Users on the other hand, may wish to bias tallys for a variety of reasons (reducing apparent popularity of products they want to buy to drive prices down, boosting popularity of items in which they or their friends have a vested interest...)

3. A large P4P system will typically involve one server and many privacy peers. e.g. the server may serve a million users, with each privacy peer serving between a hundred and ten thousand users.
4. Corruption of several privacy peers is more likely than corruption of the server and a privacy peer. From our earlier arguments, the incentives for corruption of privacy peers are very different from those for the server, but the incentives for corruption of two privacy peers are very similar(internal corruption). On the other hand, privacy peers are a much easier target for external corruption than a professionally-managed server.

For this reason, we propose a specialized threshold sharing scheme among the server and an odd number of privacy peers. If there are $2k-1$ peers, then the scheme is a $(2k, 4k-2)$-threshold secret sharing scheme where the server has $2k-1$ votes. With a threshold of $2k$, no user data is exposed unless the server is corrupted *and* at least one of the privacy peers. With that same threshold, no user data is exposed even if *all* the privacy peers are corrupted.

To achieve this threshold, we simply run the two-way, passive adversary protocol pairwise with the server as one tallyer and each of the privacy peers as the other. The same challenges c_k should be used in all cases.

Consistency checking is then automatic: At steps 3 and 4 in the user verification protocol, $\mathcal{Z}_k$ which is the commitment to the value s_k will be the same among all pairs of talliers. The user should therefore send a single $\mathcal{Z}_k$ to the server, and it should satisfy the ZKP in step 5 among all pairs of talliers.

Then we replace each decision by the privacy peer in the passive adversay protocol with the consensus of the privacy peers. Since the ZKP in step 5 demonstrates consistency of user shares, all honest talliers will reach the same decision. If there is an honest majority, the protocol will accept only legal user inputs (with high probability) and produce a correct tally.

4.4 Simulations of Typical Behavior

The system's threshold L for accepting user input is public information. The bounds we derived earlier show that *any* user vector whose L2-norm is substantially below this value 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. btw, simulation could 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 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.

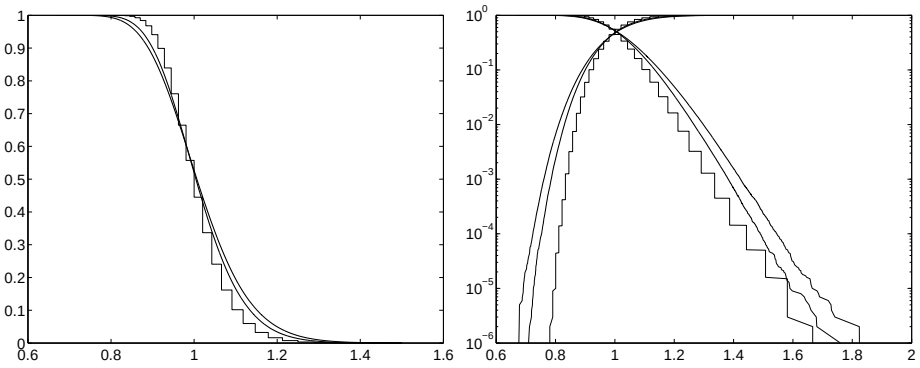

Fig. 1. (a) Linear and (b) log plots of probability of user input acceptance as a function of V_d/L for $N = 50$. (b) also includes probability of rejection. In each case, the steepest (jagged curve) is the single-value vector (case 3), the middle curve is Zipf vector (case 2) and the shallow curve is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

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 1.

4.5 Implementation and Evaluation

The protocols described in this paper are being actively developed and will be made available as a toolkit to the public shortly.[3] In the meantime we have implemented some key components of our protocols and performed some experiments to evaluate their feasibility. The protocols were implemented in Java, using a NativeBigInteger implementation from the I2P anonymous network (http://www.i2p.net/). We implemented all the components for the passive adversary model, including the ZKPs. We measured their performance on a 2.8GHz Xeon.

All tests were carried out with security parameter 1024 using El-Gamal commitments and ZKPs [1] (i.e. q is a 1024-bit number for the large field $\mathbb{Z}_q$). The number of challenges 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. Figure 2 plots prover (user client) and verifier (server or privacy peer) times for user data validation as a function of the vector size m. Prover and verifier times 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. We believe our tool will be extremely valuable for developers building real-world privacy-preserving applications.

5 Future Work

This paper shows that private vector addition with verification can be done at extremely low cost. It proposed a new model for secret-sharing called Peers-for-Privacy that should be practical in many settings. It opens the door to a variety of applications built on the vector addition primitive. In the near future, we plan to build some "middle tier" components to support those applications and add them to the toolkit. We will expand the toolkit to include not only vector addition primitives, but some common statistical aggregates such as ANOVA, SVD, correlation, and sparse factor analysis.

References

1. Cramer, R., Damgård, I.: Zero-knowledge proof for finite field arithmetic, or: Can zero-knowledge be for free? In: CRYPTO '98. Volume 1642 of Lecture Notes in Computer Science., Springer-Verlag (1998)

[3] For further information and download, please visit http://www.cs.berkeley.edu/∼duan/research/p4p.html.

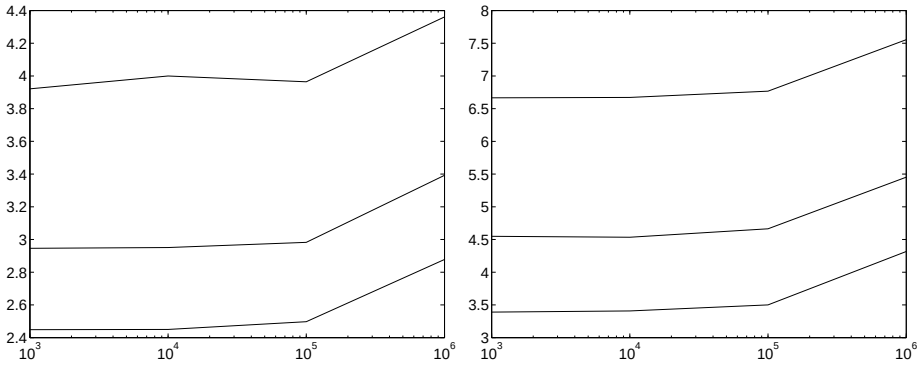

Fig. 2. (a) Verifier and (b) prover times in seconds for the validation protocol with $N = 50$, where (from top to bottom) L has 40, 20, or 10 bits. The x-axis is the vector length m.

2. Gennaro, R., Rabin, M.O., Rabin, T.: Simplified vss and fast-track multiparty computations with applications to threshold cryptography. In: PODC '98: Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing, ACM Press (1998) 101–111
3. Canny, J.: Collaborative filtering with privacy. In: IEEE Symposium on Security and Privacy, Oakland, CA (2002) 45–57
4. Canny, J.: Collaborative filtering with privacy via factor analysis. In: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Tampere, Finland, ACM Press (2002) 238–245
5. Canny, J.: Gap: a factor model for discrete data. In: SIGIR '04: Proceedings of the 27th annual international conference on Research and development in information retrieval, ACM Press (2004) 122–129
6. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project (1998)
7. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. Journal of the ACM **46**(5) (1999) 604–632
8. Duan, Y., Wang, J., Kam, M., Canny, J.: A secure online algorithm for link analysis on weighted graph. In: Proceedings of the Workshop on Link Analysis, Counterterrorism and Security at the SIAM Data Mining Conference, 2005. (2005) 71–81
9. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable secret sharing and achieving simultaneity in the presence of faults. In: Proceedings of IEEE Foundations of Computer Science. (1985) 383–395
10. Fitzi, M., Hirt, M., Maurer, U.: General adversaries in unconditional multi-party computation. In: Advances in Cryptology - ASIACRYPT 99. Volume 1716 of Lecture Notes in Computer Science., Springer-Verlag (1999) 232–246
11. Yao, A.C.C.: Protocols for secure computations. In: FOCS '82, IEEE (1982) 160–164

12. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game — a completeness theorem for protocols with honest majority. In: Proceedings of the 19th ACM Symposium on the Theory of Computing (STOC). (1987) 218–229
13. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: Proceedings of the 20th Annual ACM Symposium on Theory of Computing, STOC'88, ACM (1988) 1–10
14. Goldreich, O.: Secure multi-party computation. Working Draft (2000)
15. Beaver, D., Goldwasser, S.: Multiparty computation with faulty majority. In: Proceedings of Advances in Cryptology – CRYPTO '89. Volume 435 of Lecture Notes in Computer Science., Springer-Verlag (1989) 589
16. Goldwasser, S., Levin, L.: Fair computation of general functions in presence of immoral majority. In: Advances in Cryptology – CRYPTO '90. Volume 537 of Lecture Notes in Computer Science., Springer-Verlag (1991) 77–93
17. Evfimievski, A., Gehrke, J., Srikant, R.: Limiting privacy breaches in privacy preserving data mining. In: PODS '03: Proceedings of the twenty-second ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, New York, NY, USA, ACM Press (2003) 211–222
18. Du, W., Zhan, Z.: Using randomized response techniques for privacy-preserving data mining. In: KDD '03, New York, NY, USA, ACM Press (2003) 505–510
19. Lindell, Y., Pinkas, B.: Privacy preserving data mining. Journal of cryptology **15**(3) (2002) 177–206
20. Du, W., Han, Y., Chen, S.: Privacy-preserving multivariate statistical analysis: Linear regression and classification. In: SIAM International Conference on Data Mining. (2004) 222–233
21. Vaidya, J., Clifton, C.: Privacy-preserving k-means clustering over vertically partitioned data. In: KDD '03, New York, NY, USA, ACM Press (2003) 206–215
22. Yang, Z., Zhong, S., Wright, R.N.: Privacy-preserving classification of customer data without loss of accuracy. In: SDM 2005. (2005)
23. Kargupta, H., Datta, S., Wang, Q., Sivakumar, K.: On the privacy preserving properties of random data perturbation techniques. In: ICDM '03, Washington, DC, USA, IEEE Computer Society (2003) 99
24. Motwani, R., Raghavan., P.: Randomized Algorithms. Cambridge University Press (1995)

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

$$Pr[z < \delta NV] \leq \left(\left(\tfrac{7}{8} - \tfrac{5}{24}\delta + \tfrac{75}{288}\delta^2\right)\exp\left(\tfrac{1}{2}\delta - \tfrac{5}{12}\delta^2\right)\right)^N \qquad \text{where } 0 < \delta < 1$$

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

$$Pr[z > \delta NV] \leq \frac{E[\exp(tz)]}{\exp(\delta NV)} \tag{1}$$

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:

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

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) gives

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

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:

$$Pr[z > \delta NV] \leq \left(\tfrac{\delta}{2}\exp(1 - \tfrac{\delta}{2})\right)^N \qquad \text{where } \delta > 2 \tag{3}$$

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

$$Pr[z < \delta NV] \leq \frac{E[\exp(-tz)]}{\exp(-t\delta NV)} \tag{4}$$

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:

$$E[\exp(-tz_k)] \leq E[1 - tz_k + t^2 z_k^2/2] = 1 - tV + \tfrac{t^2}{2} E[z_k^2] \leq 1 - tV + \tfrac{3}{2} t^2 V^2 \tag{5}$$

where the last step used the moment bounds from Lemma 1. Substituting into (4) gives

$$Pr[z < \delta NV] \leq \left((1 - tV + \tfrac{3}{2} t^2 V^2) \exp(tV\delta)\right)^N \tag{6}$$

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:

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

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. Satifying 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 [24], 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:

$$E[z^q] \;=\; \sum_{r=1}^{2q} \sum_{i_1+\cdots+i_r=2q}^{1\le i_1,\ldots,i_r\le 2q} \binom{2q}{i_1, i_2, \ldots, i_r} E[s[j_1]^{i_1} \cdots s[j_r]^{i_r}] \tag{8}$$

where $1 \le j_1 < j_2 < \cdots < j_r \le m$. 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 (8) using this expansion, and using only even powers gives:

$$E[z^q] \;=\; \sum_{r=1}^{2q} \sum_{i_1+\cdots+i_r=q}^{1\le i_1,\ldots,i_r\le q} \binom{2q}{2i_1, 2i_2, \ldots, 2i_r} 2^{(-r)} d[j_1]^{2i_1} \cdots d[j_r]^{2i_r} \tag{9}$$

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

$$(d[1]^2 + \cdots + d[m]^2)^q = \sum_{r=1}^{q} \sum_{i_1+\cdots+i_r=q}^{1\le i_1,\ldots,i_r\le q} \binom{q}{i_1, i_2, \ldots, i_r} d[j_1]^{2i_1} \cdots d[j_r]^{2i_r} \tag{10}$$

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 (9) and (10), giving

$$R = 2^{-r} \binom{2q}{2i_1, \ldots, 2i_r} \Big/ \binom{q}{i_1, \ldots, i_r} \tag{11}$$

We expand this first as

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

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

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

and finally it is easy to show that $i_1! \cdots i_r! 2^r \ge 2^q$. 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 (10) by this value, we guarantee that every term in its expansion is at least as great as the coefficient in equation (9). Or in other words,

$$E[z^q] \ \leq \frac{(2q)!}{q!2^q} V^q = (1 \cdot 3 \cdot 5 \cdots (2q-1)) V^q \leq (qV)^q \qquad \text{QED} \quad (12)$$