Zero-knowledge Test of Vector Equivalence and Granulation of User Data with Privacy

Yitao Duan and John Canny

***Abstract*— This paper introduces a new framework for privacy preserving computation to the granular computing community. The framework is called P4P (Peers for Privacy) and features a unique architecture and practical protocols for user data validation and vector addition based computation. It turned out that many non-trivial and non-linear computations can be done using an iterative algorithm with vector-addition aggregation steps. Examples include voting, summation, SVD, regression, and ANOVA etc. P4P allows them to be carried out while preserving users privacy. To demonstrate its application in granular computing, we present two efficient protocols that test the equality of user vectors in zero-knowledge. These protocols can be used to perform granulation, which is a fundamental task of granular computing, in a privacy-preserving manner. They can also be of independent interest for other fields such as data mining as well.**

***Index Terms*— Privacy, zero-knowledge protocol, equivalence test, granulation.**

I. Introduction

GRANULATION of the universe is one of the fundamental issues in granular computing (GrC). It involves decomposing the universe into subsets or clustering individual objects into classes. The subsets or classes are called granules. A granulation may consists of a family of either disjoint or overlapping granules. It is the first step towards treating the world in a GrC manner.

A granulation must use some criteria that determines whether two objects should be put into the same granule. A simple scheme can be based on equivalence relation. Let U be a finite and non-empty set of objects called the universe and $E \subseteq U \times U$ an equivalence relation on U. E divides U into a family of disjoint subsets called the partition of the universe induced by E. Two objects are equivalent if they are in the same subset.

An equivalence relation E on U essentially defines a granulation. The equivalence relation itself can be determined by knowledge about the objects. Such knowledge can be provided by schemes such as information table [1], [2]. An information table is defined as the tuple:

$$(U, At, \{V_a | a \in At\}, \{I_a | a \in At\})$$

where At is a finite set of attributes or features, V_a is a set of values for each attribute $a \in At$, and $I_a : U \to V_a$ is an information function for each attribute $a \in At$ such that it maps an object in U to exactly one value in V_a.

In this scheme, given a subset $A \subseteq At$, we can define an equivalence relation E_A as:

$$xE_A y \Leftrightarrow \forall a \in A,\, I_a(x) = I_a(y)$$

In other words, x and y are indiscernible, with respect to all attributes in A, if and only if they have the same value for every attribute in A.

A. Granulation and Privacy

In many situations, the information table contains users' private information. For example, in health care applications, the information table can be a database containing users' gender, ethnic, age, medical history etc. In this case it is extremely valuable for many health care providers to granulate the data accurately to gain statistical insights and mine useful information. Yet it is also extremely detrimental to user privacy if the raw information is exposed to a single provider.

We provide a private computation framework called Peers for Privacy (P4P) to address this dilemma. P4P was first introduced in [3]. It includes efficient and privacy-preserving protocols for performing user data validation and many useful computations. P4P offers a practical solution to many application such as privacy-preserving data mining etc. In this paper we show that P4P also has great potential in preserving user privacy in granular computing as well. Concretely, we introduce two efficient zero-knowledge protocols that test the equality of two user vectors or some of their elements. These protocols can be used to implement the equivalence relation mentioned earlier. Therefore we provide a practical privacy-preserving solution for granulation of user data. By doing so we wish to bring to the attention of the GrC research community the importance of user privacy and encourage efforts to search for practical solutions for preserving it.

The rest of the paper is organized as follows. Section II examines previous solutions and other related research. In Section III we give a brief overview of the P4P framework. Section IV states our model and assumptions. Section V summarizes some cryptographic tools that our protocols use. In VI we give detailed description of our protocols and their security proofs. Finally we discuss current implementation issues and future directions in Section VII.

II. Related Work

Zadeh first introduced the concept of information granulation in the context of fuzzy set in 1979 [4]. Almost two

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

Both authors are with the Computer Science Division, University of California at Berkeley, 387 Soda Hall, UC Berkeley, Berkeley, CA 94720-1776 (emails: {duan, jfc}@cs.berkeley.edu).

decades later, Lin [5] coined the term granular computing to label a new and fast-growing field of multi-disciplinary research that includes fuzzy and rough set theories [1], [4], [6], data mining [7], intelligence systems [8] etc. Granulation by equivalence relation can be found in many works such as [9], [10].

The tasks we describe in this paper can be carried out using Secure multiparty computation (MPC) protocols. MPC dates back to Yao [11] and Goldreich et al. [12]. Important works include [13], [14], [15], [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 due to their heavy use of public-key operations or zero-knowledge proofs (ZKP).

[3] introduced a new paradigm for performing computations based on vector addition. In that setting, each user inputs a vector and the computation is carried out in rounds, each of which only involves vector addition. As [3] and other works (e.g. [17]), showed, many non-trivial and non-linear computations can be done using an iterative algorithm with vector-addition aggregation steps. Examples include voting, summation, SVD, regression, and ANOVA etc. The architecture introduced in [3] is called Peers for Privacy (P4P). It allows the above algorithms to be computed in much more efficiently than generic MPC protocols. In addition, [3] also provided a very efficient protocol that verifies, in zero-knowledge, that the user input is valid. Our work in this paper builds upon [3]. We introduce new protocols to the P4P framework and show that granulation of user data can also be performed efficiently with privacy in the same setting.

The task our protocols perform bears some similarity with the zero-knowledge proofs that two commitments/encryptions encode the same number (e.g. [18], [19] and the disjunctive proof of equality of plaintext (DISPEP) and proof of equality of plaintext (PEP) of [20]). Both are testing the equivalence of obfuscated data without revealing the clear text. However, there is a subtle but fundamental difference: ours is not a *proof* system in that there does not exist a prover who knows the pre-images or the decryption key, as is required by the ZKPs. Rather, ours is a private *computation* task. The techniques developed in e.g. [18], [19], [20] cannot be directly applied directly. This will be further elaborated in Section VI-A.

III. P4P: An Overview

In P4P, we assume there is 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. In contrast to previous work, privacy peers are assumed to belong to users in the community and each privacy peer should service a small number of users. 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.

This architecture is a hybrid of client-server and P2P. On one hand, the server shares the bulk of the computation and storage, and also synchronizes the protocol. This allows us to take advantage of its large comoutation/storage capacity and high availability. It also leads to practical, efficient protocols that are not possible with fully distributed architecture. On the other hand, the peers also participate in the computation, and offload information, thus trust, from the server and provide privacy. Practical P2P systems such as Gnutella and Napster showed that the existence of altruistic users who provide services to others is a pervasive phenomenon in communities. Therefore it is not a problem to find such users to volunteer as privacy peers. In certain cases, instead of using peers from the users, it may be feasible to distribute the computation among service providers. For example, two hospitals may wish to mine data from user records. P4P can be used to support this type of computation by letting one of the service providers to assume the role of privacy peer in the protocol and the result is that both hospitals learn the (aggregate) final computation but neither learns anything about users private data.

The security of a P4P system is based upon the assumption that the server and the privacy peers won't be corrupted at the same time (see adversary model later). This is realistic in many situations. The key observation is, the server is typically well-protected (at least cooperations spend large amount of money trying) therefore the server-peer pair is immune against external attacks. And we argue that the server and the privacy peers do not have incentive for collusion because there is a mutual *distrust* among them and the risk of discovery is high: colluding between the server and the privacy peer requires them to exchange data, and both will be aware of the cheating. Neither can trust the other not to expose the cheating (some exposure maybe by accident). This assumption allows us to leverage the differences between the players and construct efficient protocols that best utilize their individual advantages: Essentially our system relies on the server for defending against *outside* attacks and uses the privacy peers to protect user privacy against a curious/malicious server.

One of the major advantages of P4P architecture is that it allows the main computation to be executed in the "normal"-sized field where each integer fits into a single memory cell. Big integer field is used only for verification which involves only a small number (constant or $O(\log m)$ where m is the size of user data) of big integer operations. In contrast, generic MPC protocols (e.g. [13], [14], [15], [16], [21]) work in the big field all the time. In a typical computer today there is a six order of magnitude difference between the big integer cryptographic operations (order of milliseconds) and regular arithmetic operations (fraction of a nano-second) and P4P provides a practical solution for large scaled private computation tasks. Our new protocols introduced in this paper follow similar approach, i.e. main computation is small field and small number of big integer operations for verification, thus preserves its efficiency.

Similar to [3], we describe our protocols as 2-way multi-party computations carried out between the server and a privacy peer who are referred collectively to as *talliers*. In this paper we denote the server T_1 and the privacy peer T_2.

IV. Preliminaries

We assume all n users have access to secure channels with the server and the privacy peer(s). Let ϕ be a small integer [1] (e.g. 32 or 64 bits). Our goal is to support "normal"-sized integer (or fixed-point) arithmetics. This is what most application needs and the arithmetics are extremely efficient when each integer fits into a single memory cell. To provide information-hiding, we use a $(2, 2)$-threshold secret sharing to embed this integer range in the additive group of integers modulo ϕ. To support signed values, which is what most applications require, 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 field.

Let $d_i \in \mathbb{Z}_\phi^m$ be an m-dimensional data vector for user i. We use $x \leftarrow_R X$ to denote the assignment to x an element uniformly randomly selected from a set X. Throughout this paper we use $u \cdot v$ to denote the inner product of two vectors u and v.

A. Basic P4P Computation

The goal of the main P4P protocol is to compute the sum of all user vectors (As we mentioned earlier, many useful computation can be decomposed into such steps. Please see [3]). It is performed as follows:

Assume $Q = \{1, \ldots, n\}$ is the initial set of qualified users. The basic computation in P4P 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 described in [3]. 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 $\mu + \nu \mod \phi$.

Both the correctness and the privacy of the basic computation are demonstrated in [3].

B. Adversary Model

In this paper, we only consider the "passive" adversary model of [3] which is actually a mixed model: an adversary is allowed to actively corrupt any number of users, causing them to deviate arbitrarily from the specified protocol, 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. The protocols can be patched with techniques such as ZKP or consensus to deal with actively corrupted tallier.

V. Tools

Our protocols use some standard cryptographic primitives for homomorphic computation. They have appeared elsewhere, see e.g. [18], [19]. Here we summarize their key properties.

A. Homomorphic Commitment

Our protocols use a cryptographic primitive called homomorphic commitment. A commitment allows a prover to give a verifier a commitment to a secret number. The verifier should not be able to compute any information about the secret from the commitment. The prover can later open the commitment by revealing the secret, and any auxiliary random bits accompanying this commitment, and the verifier can verify that the commitment indeed "contains" the secret. If the prover lies about the secret, it will be detected.

Denote by $\mathcal{C}(a, r)$ a commitment to an integer a with randomness r. The scheme should have the standard "hiding" and "binding" properties as defined in the literature, i.e. it is (cryptographically or information-theoretically) hard to either determine a given $\mathcal{C}(a, r)$ or find $a' \neq a$ and r' such that $\mathcal{C}(a, r) = \mathcal{C}(a', r')$. In addition, the commitment is homomorphic if the following holds: Given $\mathcal{A}_1 = \mathcal{C}(a_1, r_1)$ and $\mathcal{A}_2 = \mathcal{C}(a_2, r_2)$, there exists some r such that $\mathcal{A}_2\mathcal{A}_2 = \mathcal{C}(a_1 + a_2, r)$.

Many commitment schemes have this homomorphism property. We use Pedersen's discrete log based scheme [18] for it admits an efficiency ZKP for equivalence. Let p and q be two large primes such that $q|p-1$. Let $\mathbb{Z}_p^*$ denote the multiplicative group of integers modulo the prime p. We use G_q to denote the unique subgroup of $\mathbb{Z}_p^*$ of order q. The discrete logarithm problem is assumed to be hard in G_q. Let g and h be two generators of G_q such that $\log_g h$ is unknown to anyone. [2] A commitment to a is computed as $\mathcal{C}(a, r) = g^a h^r \mod p$ where $r \leftarrow_R \mathbb{Z}_q$. From now on, $\mathcal{C}(a, r)$ will denote such Pedersen commitment function. We will omit the randomness r from the notation, and simply write $A = \mathcal{C}(a)$, if it is not necessary to identify it.

B. Multiply by A Constant

Let A be commitment to $a \in \mathbb{Z}_q$ and $c \in \mathbb{Z}_q$ a constant. One can easily obtain a commitment to $d = ac$ by

$$D = A^c \mod p$$

This follows immediately from homomorphism. However, only the prover who knows how to open A can open D.

C. 3-way Commitment and ZKP

For integer $c \in \mathbb{Z}_q$, a 3-way commitment, denoted $(0, \pm c)$-commitment, is a commitment to one of $0, c$ or $-c$. For such commitment we can construct an efficient zero-knowledge proof.

Let $a \in \{0, c, -c\}$ be the number to be committed to. The prover computes two commitments B and C such that

$$\begin{aligned} B = \mathcal{C}(0), C = \mathcal{C}(0) &\text{ if } a = 0 \\ B = \mathcal{C}(1), C = \mathcal{C}(0) &\text{ if } a = c \\ B = \mathcal{C}(0), C = \mathcal{C}(1) &\text{ if } a = -c \end{aligned}$$

[1] It does not have to be a prime if the computation only involves addition.

[2] A generator of G_q can be easily found by selecting an element $a \leftarrow_R \mathbb{Z}_p^*, a \neq 1$ and testing if $a^q = 1$, since any element $a \neq 1$ in G_q generates the group. g and h can chosen by the two talliers using a coin-flipping protocol.

The prover also provides zero-knowledge proofs that both B and C encode either 0 or 1 using the *bit commitment* proof of [19]. Finally, the commitment to a is simply

$$A = B^c C^{-c} \mod p$$

and the 3-way $(0, \pm c)$-commitment proof consists of (B, C) and their bit commitment proofs.

To verify the proof, a verifier checks that $A = B^c C^{-c} \mod p$ and that the bit commitment proofs are valid. If all these verifications are successful, the verifier accepts the proof. Otherwise it rejects it.

It is easy to show that only when A encodes one of $\{0, c, -c\}$ will the verifier accept the proof. And it is zero-knowledge due to the hiding property of the commitment scheme. The proof can also be made non-interactive by hashing the verifiers response.

VI. ZK Test of Equivalence

Equivalence relation can be defined by equality. In this section we introduce two equality test protocols. One tests a single element, the other the whole user vector. Both enjoy the following:

1) No information about user data is leaked;
2) Only a small number of public key operations are involved;
3) Users do not need to be involved after the initial data input stage of the main P4P protocol.

A. Equality Test of A Single Element

Let a_i be the element of user i that defines the equivalence relation E which partitions U. Recall that the two shares of a_i, denoted a_{i1} and a_{i2}, are already sent to T_1 and T_2, respectively, in the main P4P protocol. The goal is, given two user indexes i, j, to determine whether $a_i = a_j$.

This task is not as trivial as it appears. It is true that, with homomorphic commitment, verifying whether two commitments contain equal numbers in zero-knowledge is easy provided there is a prover holding the pre-images [18], [19]. Our setting, however, is a different model. Namely, in P4P, there is no such a prover who knows both numbers. Instead, each tallier holds a share of each number. Collaboratively they want to determine whether the two are equal. In other words, ours is not a zero-knowledge proof task. Rather, it is a zero-knowledge test or a two-party computation problem computing a boolean function that returns 1 if $a_i = a_j$ and 0 otherwise. This, of course must be done without leaking any information about the numbers.

The difficulty in applying existing ZKP lies in the fact that the definition of zero-knowledge in a ZKP system protects prover's privacy only when the statement is true. To see this, let $A_i = \mathcal{C}(a_i, r_i)$ and $A_j = \mathcal{C}(a_j, r_j)$ where the prover knows (a_i, r_i, a_j, r_j). The technique for proving A_i and A_j contain the same number involves the prover revealing $\delta = r_i - r_j$ and the verifier checking if $A_i A_j^{-1} = h^\delta$ holds [18], [19]. The problem is, if $a_i \neq a_j$, revealing δ also reveals some information about $a_i - a_j$. Namely once the random mask is exposed, one can obtain $g^{a_i - a_j} = A_i A_j^{-1} h^{-\delta} \mod p$. And when $a_i - a_j$ is small, it is easy to recover it. In a standard ZKP setting, the prover can just admit $a_i \neq a_j$ when it is the case, thus avoiding leaking information. But this trick is not possible in P4P where no such prover exists. The same is true for the DISPEP and PEP [20] techniques which use ElGamal encryption.

We develop the following protocol to address this issue. To enable equality test, the users first escrow some information with the talliers. Specifically, for $j = 1, 2$, user i and the talliers perform the following:

1) User i computes $A_{ij} = \mathcal{C}(a_{ij}, r_{ij})$ where $r_{ij} \leftarrow_R \mathbb{Z}_q$. She also prepares a 3-way $(0, \pm\phi)$-commitment B_i with pre-image $b_i = a_i - (a_{i1} + a_{i2})$ and randomness s_i. She then shares both b_i and s_i:

$$s_{i1} \leftarrow_R \mathbb{Z}_q, \quad s_{i2} = s_i - s_{i1} \mod q$$
$$b_{i1} \leftarrow_R \mathbb{Z}_q, \quad b_{i2} = b_i - b_{i1} \mod q$$

She sends $(A_{ij}, r_{ij}, b_{ij}, s_{ij})$ to T_j and broadcasts to both talliers $B_{ij} = \mathcal{C}(b_{ij}, r_{ij})$, B_i and its corresponding 3-way $(0, \pm\phi)$-commitment proof.

2) T_j verifies that $A_{ij} = \mathcal{C}(a_{ij}, r_{ij})$ and $B_{ij} = \mathcal{C}(b_{ij}, r_{ij})$. Both talliers verify that $B_i = B_{i1} B_{i2}$ and that the $(0, \pm\phi)$-commitment proof is valid. If any of the verification fails, user i is excluded from the computation.

The above is executed in the user input stage together with data validation [3]. It is the only stage involving user interaction. The actual test can be carried out between the two tallier afterwards. The users do not have to be online at all times.

EQUALITY-TEST:

Without loss of generality, suppose we want to check if $a_1 = a_2$. In the following description, $j \in \{1, 2\}$.

1) Both talliers compute

$$A_1 = A_{11} A_{12} B_{11} B_{12} \mod p$$
$$A_2 = A_{21} A_{22} B_{21} B_{22} \mod p$$

and $\Delta = A_1 A_2^{-1} \mod p$.

2) T_j computes $\delta_j = (r_{1j} + s_{1j}) - (r_{2j} + s_{2j}) \mod q$.

3) T_j generates a random number $k_j \leftarrow_R \mathbb{Z}_q^* \setminus \{1\}$ and computes $\Delta_j = \Delta^{k_j} \mod p$, $H_j = h^{k_j} \mod p$.

4) The two talliers exchange (Δ_j, H_j). If T_i finds $H_j = h, j \in \{1, 2\}, j \neq i$, he aborts the protocol.

5) T_1 publishes $\bar{H}_1 = (H_1 H_2)^{\delta_1}$ which is $h^{k\delta_1}$, and T_2 publishes $\bar{H}_2 = (H_1 H_2)^{\delta_2}$ which equals to $h^{k\delta_2}$, where $k = k_1 + k_2 \mod q$.

6) Both talliers verify if

$$\Delta_1 \Delta_2 \equiv \bar{H}_1 \bar{H}_2 \pmod p \qquad (1)$$

If it holds, then $a_1 = a_2$. Otherwise $a_1 \neq a_2$.

Theorem 1: The above protocol correctly tests if $a_1 = a_2$. Furthermore it does not leak any information about user data.

Proof: (Sketch) The completeness of the protocol follows the homomorphism property. Note that Δ computed in Step 1 is a commitment to $a_1 - a_2$ with randomness $\delta_1 + \delta_2$. It follows that $\Delta_1\Delta_2$ in Equation 1 is a commitment to $k(a_1 - a_2)$ with randomness $k(\delta_1 + \delta_2)$. If $a_1 = a_2$, $\Delta_1\Delta_2$ should open to 0, and Equation 1 should hold.

The soundness is guaranteed by the binding property of the commitment and the fact that the probability of $k_1 + k_2 \equiv 0 \bmod q$ is very small (only $\frac{1}{q}$).

To show that the protocol is zero-knowledge, we construct a simulator that takes as inputs the corrupted player's data, the public information, and the final output, and interacts with the adversary in a simulated execution of the protocol. We need to show that this execution is indistinguishable to the (polynomial time) adversary.

Without loss of generality, let us suppose tallier T_1 is corrupted. Note that Δ is common inputs to both talliers computed from users' public commitments. The simulator only needs to produce the rest of the conversation.

Let $\bar{\mathbb{Z}}_q^* = \mathbb{Z}_q^* \setminus \{1\}$. The view of the adversary (i.e. T_1) during a real execution of the protocol is: [3]

$$VIEW_{Real}^{T_1} = [h, k_1, \delta_1, h^{k_2}, h^{(k_1+k_2)\delta_2}]_{k_1,k_2 \in \bar{\mathbb{Z}}_q^*, \delta_1, \delta_2 \leftarrow_R \mathbb{Z}_q}$$

For a non-passing execution (i.e. one that outputs $a_1 \neq a_2$), the simulator just generates two random numbers $k_2' \in \bar{\mathbb{Z}}_q^*, \delta_2' \leftarrow_R \mathbb{Z}_q$ and uses k_2', δ_2' in place of k_2, δ_2 in the protocol. Clearly the transcript of the simulation follows the same distribution as that of an actual execution of the protocol and only with negligible probability will the simulated protocol (incorrectly) output $a_1 = a_2$.

For a passing test (i.e. one execution that outputs $a_1 = a_2$), the simulator works as follows:

It generates $k_2' \leftarrow_R \bar{\mathbb{Z}}_q^*$ and computes Δ_2 and H_2 as specified by the protocol but with k_2' in place of k_2. It then computes the rest of the information as required by the protocol as

$$\begin{aligned} \bar{H}_2 &= \Delta_1\Delta_2/\bar{H}_1 \mod p \\ &= h^{(k_1+k_2')\delta_2} \mod p \end{aligned}$$

The adversary's view in the simulated execution is then

$$VIEW_{Sim}^{T_1} = [h, k_1, \delta_1, h^{k_2'}, h^{(k_1+k_2')\delta_2}]_{k_1,k_2' \in \bar{\mathbb{Z}}_q^*, \delta_1, \delta_2 \leftarrow_R \mathbb{Z}_q}$$

Clearly this distribution is identical to $VIEW_{Real}^{T_1}$, the adversary's view in a real execution.

And finally the final output reveals $g^{k(a_1-a_2)}$, not $g^{a_1-a_2}$. When $a_1 \neq a_2$, this quantity leaks no information about either a_1 or a_2, or their difference. This is because when $a_1 \neq a_2$, the difference is in the multiplicative group $\mathbb{Z}_q^*$ and has an inverse mod q. For any given value c and a_1, a_2, there is a $k = c(a_1 - a_2)^{-1} \mod q$ such that $k(a_1 - a_2) \equiv c \bmod q$. In other words, $a_1 - a_2$ is equally likely to take any values in $\mathbb{Z}_q^*$ even $k(a_1 - a_2)$ is revealed. ■

The use of B_j in the protocol is to deal with modular reduction. Note that a_i, a_{i1} and a_{i2} are all in the small field $\mathbb{Z}_\phi$. In order for the shares a_{i1} and a_{i2} not to leak any information about a_i, they should be computed as $a_{i1} \leftarrow_R \mathbb{Z}_\phi, a_{i2} = a_i - a_{i1} \mod \phi$. This means $b_i = a_i - (a_{i1} + a_{i1})$ can be 0 or $\pm\phi$. Using B_j in the protocol is to correct the modular reduction and obtain the actual commitment to a_i. Also note that the sharing of b_i is in the big field $\mathbb{Z}_q$ and there is no modular reduction problem here because of commitment is in G_q, the cyclic group of order q.

B. Equality Test of the Whole Vector

Equality test of the whole vector can be done via m element tests introduced in Section VI-A. However, this involves $O(m)$ public key operations and is not practical for large m. The following protocol, in contrast, requires only $O(1)$ public key operation and is very efficient.

Our protocol uses similar ideas as in [3], i.e. instead of checking every element, it checks the projections of the vectors on a random challenge vector. We show later that, if two vectors have equal projections on these directions then, with high probability, the two two vectors are equal.

Suppose we are checking if $d_1 = d_2$. Recall that after the input and validation stage in [3], T_1 holds u_1, u_2 and T_2 has v_1, v_2 such that $d_1 = u_1 + v_1 \mod \phi$ and $d_2 = u_2 + v_2 \mod \phi$.

EQUALITY-TEST-V:

1) T_1 and T_2 generate a random challenge vectors $c \leftarrow_R \mathbb{Z}_q^m$ using the a common random seed r they agreed upon with some protocol (e.g. [3]).
2) T_1 computes $x = c \cdot (u_1 - u_2) \mod \phi$ and T_2 computes $y = c \cdot (v_1 - v_2) \mod \phi$.
3) T_1 commits to x with $X = \mathcal{C}(x, \delta_1), \delta_1 \leftarrow_R \mathbb{Z}_q$. Similarly T_2 commits to y: $Y = \mathcal{C}(y, \delta_2), \delta_2 \leftarrow_R \mathbb{Z}_q$. The two exchange X, Y and compute the following 3 numbers:
$$\begin{aligned} Z_1 &= XY \mod p \\ Z_2 &= XYg^{\phi} \mod p \\ Z_2 &= XYg^{-\phi} \mod p \end{aligned}$$
4) For $i = 1, 2, 3$, the two talliers run steps 3 to 6 of the single element equality test protocol, with Δ replaced by Z_i. If any of the three runs outputs positive result (meaning the two numbers being tested are equal), output $d_1 = d_2$. Otherwise output $d_1 \neq d_2$.

Theorem 2: Let O be the output of protocol EQUALITY-TEST-V. Let $O = 1$ if the protocol concludes that $d_1 = d_2$ and 0 otherwise. If $d_1 \neq d_2$, the probability that the protocol (incorrectly) outputs 1 is at most

$$Pr(O = 1) \leq \frac{1}{\phi}$$

Proof: Let $\delta = d_1 - d_2$. Note that $c \cdot \delta = c \cdot ((u_1 + v_1) - (u_2 + v_2)) \mod \phi = x + y \mod \phi$. Consider the N-dimensional vector space W over $\mathbb{Z}_\phi$. W has ϕ^N elements.

[3] Note that we omit the commitments $(A_1, A_2, \Delta, \Delta_1)$ since they are public and randomly uniformly distributed in G_q (recall that Pedersen's commitment scheme is information-theoretic hiding [18]).

For any $\delta \in \mathbb{Z}_{\phi}^{N}$, there are at most ϕ^{N-1} vectors in this space that are orthogonal to it, i.e. those on a hyperplane V which is a codimension-1 vector subspace of W and has size ϕ^{N-1}. Since δ is not known to either tallier and the c is randomly drawn, we have

$$Pr(c \cdot \delta = 0) \leq \frac{1}{\phi}$$

Now we show that the protocol actually tests on $c \cdot \delta$. Note that $Z_1 = XY \mod p$ encodes $x + y$. And $c \cdot \delta = x + y \mod \phi = x + y + b$ where b is one of 0 or $\pm\phi$. Clearly $x + y \mod \phi = 0$ is equivalent to one of $x+y, x+y+\phi$ and $x+y-\phi$ is 0, which is what the protocol tests. ■

In terms of privacy, note that the talliers compute the projection of the user vectors on a random direction and then test equality on the projections. Due to the zero-knowledge property of the EQUALITY-TEST protocol, this protocol does not leak information either.

In practice, ϕ is typically 2^{32} or 2^{64}, so that the number fits into a machine word. This gives a failure probability of 2.4×10^{-10} or 5.4×10^{-20}. This should be enough for most applications.

The protocols described in Section VI-A and VI-B represent two ends of a spectrum, i.e., partitioning the universe by the equivalence relation E_A defined by a single element and the whole vector, respectively. If we want to test only certain elements of the user vectors, we can use the second protocol but with challenge vectors having 0's at certain places to mask out irrelevant elements.

VII. Conclusion and Future Work

In this paper we demonstrated possible applications of a private computation framework in GrC. We presented efficient protocols that can be used to granulate user data while preserving user's privacy. These protocols can also support meaningful applications by themselves. For example, consider a location-based service (LBS) application. A user may wish to find all her friends near-by. The user vector in this case will be locations. The user sets the element corresponding to her current location to 1 and the rest 0's. She then uploads the vector to the system. The system can use the vector equality test protocol presented in this paper to match, among her friends, those who have equal location vectors.

The P4P framework and the protocols described in this paper are being actively developed. Please visit http://www.cs.berkeley.edu/∼duan/research/p4p.html for further information. We have measured the performance of the key components of the framework on a 2.8GHz Xeon and the numbers were reported in [3]. The results showed that the protocols are indeed practical for large systems (e.g. with $m = 10^6$).

In the near future, we plan to build some "middle tier" components to support more concrete applications. We will include not only vector addition primitives, but some common statistical aggregates such as ANOVA, SVD, correlation, and sparse factor analysis. We believe it will be a valuable tool for developers in areas such as GrC, data mining and others, to build privacy preserving real-world applications.

References

[1] Z. Pawlak, *Rough Sets: Theoretical Aspects of Reasoning about Data*. Norwell, MA, USA: Kluwer Academic Publishers, 1992.

[2] Y. Y. Yao and N. Zhong, "Granular computing using information tables," pp. 102–124, 2002.

[3] J. F. Canny and Y. Duan, "Practical private computation of vector addition-based functions or: Can privacy be for free?" EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2006-12, February 8 2006. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-12.html

[4] L. A. Zadeh, "Fuzzy sets and information granularity," pp. 3–18, 1979.

[5] T. Y. Lin, "Granular computing, announcement of the bisc special interest group on granular computing," 1997.

[6] Z. Pawlak, "Reasoning about data - a rough set perspective," in *RSCTC '98: Proceedings of the First International Conference on Rough Sets and Current Trends in Computing*. London, UK: Springer-Verlag, 1998, pp. 25–34.

[7] T. Y. Lin, "Granular computing on binary relations," in *TSCTC '02: Proceedings of the Third International Conference on Rough Sets and Current Trends in Computing*. London, UK: Springer-Verlag, 2002, pp. 296–299.

[8] A. Skowron, "Toward intelligent systems: Calculi of information granules," in *Proceedings of the Joint JSAI 2001 Workshop on New Frontiers in Artificial Intelligence*. London, UK: Springer-Verlag, 2001, pp. 251–260.

[9] Y. Y. Yao, "A partition model of granular computing," *LNCS Transactions on Rough Sets*, vol. 1, pp. 232–253, 2004.

[10] Y. Y. Yao and C.-J. Liau, "A generalized decision logic language for granular computing," in *FUZZ-IEEE'02: The 2002 IEEE World Congress on Computational Intelligence*, 2002, pp. 1092–1097.

[11] A. C.-C. Yao, "Protocols for secure computations," in *FOCS '82*. IEEE, 1982, pp. 160–164.

[12] O. Goldreich, S. Micali, and A. Wigderson, "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, pp. 218–229.

[13] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, STOC'88*. ACM, May 2–4 1988, pp. 1–10.

[14] O. Goldreich, "Secure multi-party computation," Working Draft, 2000. [Online]. Available: citeseer.nj.nec.com/goldreich98secure.html

[15] D. Beaver and S. Goldwasser, "Multiparty computation with faulty majority," in *Proceedings of Advances in Cryptology – CRYPTO '89*, ser. Lecture Notes in Computer Science, vol. 435. Springer-Verlag, 1989, p. 589.

[16] S. Goldwasser and L. Levin, "Fair computation of general functions in presence of immoral majority," in *Advances in Cryptology – CRYPTO '90*, ser. Lecture Notes in Computer Science, vol. 537. Springer-Verlag, 1991, pp. 77–93.

[17] J. Canny, "Collaborative filtering with privacy," in *IEEE Symposium on Security and Privacy*, Oakland, CA, May 2002, pp. 45–57. [Online]. Available: http://citeseer.nj.nec.com/canny02collaborative.html

[18] T. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology – CRYPTO '91*, ser. Lecture Notes in Computer Science, vol. 576. Springer-Verlag, 1991, pp. 129–140.

[19] R. Cramer and I. Damgård, "Zero-knowledge proof for finite field arithmetic, or: Can zero-knowledge be for free?" in *CRYPTO '98*, ser. Lecture Notes in Computer Science, vol. 1642. Springer-Verlag, 1998.

[20] J. Markus and J. Ari, "Millimix: Mixing in small batches," Tech. Rep., 1999.

[21] R. Gennaro, M. O. Rabin, and T. Rabin, "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, pp. 101–111.