Privacy-Preserving Distributed Algorithm for Sparse Vector Sum

Sheng Zhong

Abstract

We study the following problem: B_0 and B_1 each has a sparse input vector V_0 and V_1; for each j we need to decide whether $B_0[j] + B_1[j] > t$. We give a privacy-preserving algorithm, in which B_0 and B_1 do not need to reveal any information about their input vectors to each other, except the output of algorithm. Our algorithm is highly efficient.

1 Introduction

Suppose there is a neighborhood with two local banks. Many of the college students in the neighborhood need to apply for student loans to support their study, and both of the two local banks provide student loans. However, the banks would like to make sure that each student gets no more than ten thousand dollars in total for loans from both of them. Clearly, this would be a trivial problem if the two local banks were allowed to exchange private information about their customers, like how much a specific student has got from each of them. However, customers do not want their banks to reveal their private information to anybody, including to the other local bank. Can we solve this problem in a privacy-preserving way?

Formally, denote by B_0 and B_1 the two local banks involved. Each bank B_i has an n-dimensional input vector V_i, where $V_i[j]$ is the total amount of money (in thousand dollars) bank B_i has loaned to customer j. We ask whether there is a privacy-preserving distributed algorithm that decides, for each customer j,

Received by the editors March 2008.
Communicated by A. Weierman.
2000 *Mathematics Subject Classification* : 11T71; 94A60.
Key words and phrases : Cryptography; Distributed Algorithm; Sparse Vector.

whether $V_0[j] + V_1[j] > t$, where $t = 10$ (thousand dollars) is the threshold for total amount of loans. Here by "privacy-preserving" we mean each bank B_i should not reveal any information about V_i to the other bank B_{1-i}, including whether $V_i[j] = 0$ for a specific customer j (i.e., whether bank B_i has loaned any money to customer j at all.)

Since each bank has no knowledge about the customers of the other bank, we assume that customers are identified using their social security numbers. Consequently, each input vector V_i is *sparse* in that, for almost all social security numbers j, $V_i[j] = 0$ (i.e., customer j has never applied for any loan from the local banks). Note that, although for almost all j, $V_0[j] = V_1[j] = 0$, the banks can not reduce the dimension of their input vectors, e.g., by jointly computing the set of customers to whom they have loaned. This is because such a customer set allows each bank to learn partial information about the other bank's input vector. For example, if bank B_0 sees customer j in the set of customers who have got loans, and it knows that it has never loaned any money to customer j, then it learns that bank B_1 has loaned to customer j. For privacy protection of customers, we do not allow this to happen. We require that, all each bank can learn from running the distributed algorithm for sparse vector sum is whether $V_0[j] + V_1[j] > t$ for each j; anything else it knows after running the algorithm is implied by the above result and its a priori knowledge before running the algorithm. Consequently, we have to design a privacy-preserving algorithm for sparse input vectors.

1.1 Related Work

The above problem of *sparse vector sum* can be viewed as an extension of Yao's millionaire problem [9] to sparse vectors. However, it is very challenging to design an efficient solution to this problem. In particular, consider a naive solution which runs n instances of Yao's solution to the millionaire problem in parallel. Since n is very large and the input vectors are sparse, the naive solution is very expensive, spending a huge amount of the time on j such that $V_0[j] = V_1[j] = 0$. The objective of this paper is to develop a fast solution that has reasonable computational overhead even for really large n.

Since this problem is a special case of *secure multi-party computation*, there exist many general-purpose solutions that can be applied to this problem. In particular, Yao [10] and Goldreich, et al. [8] presented completeness theorems in the computationally bounded model. In the computationally unbounded model, Ben-Or, et al. [1] and Chaum, et al. [2] gave similar theorems. We stress that, as observed by Goldreich [7], these general-purpose results are expensive in computational and communication overheads. Thus, the target of our work is to provide an *efficient* solution to this problem.

We note that Cramer and Damgård [3] studied secure distributed linear algebra and it is also related to our problem. However, our problem does not fall into the class of problems they solved in [3].

Another thread of related research is the study of secure set intersection (see, *e.g.*, [4, 5], among others). We emphasize that, although one might be able to reduce our problem of sparse vector to the problem of secure set intersection, the resulting instance of secure set intersection problem would have a very large input size (*e.g.*, $\Omega(nt)$). Hence, it would be much less efficient to solve our problem using such reductions.

2 Technical Preliminaries

Before we present our solution to the problem of sparse vector sum, we first give a brief review of the formal definition of privacy and the cryptographic tool we use–ElGamal encryption.

2.1 Formal Definition of Privacy

Our privacy definition comes from an adaptation of the standard definition of privacy for cryptographic protocols in the *semi-honest model*. Here, the semi-honest model is a standard cryptographic model in which each involved party works exactly as specified in the protocol/algorithm but may attempt to derive extra information. (For details about this model, see [7].)

In our privacy definition given below, we need to use a function $SVS()$ to denote the correct output of our sparse vector sum problem. Formally, we let $SVS(V_0, V_1)$ be an n-dimensional vector such that

$$SVS(V_0, V_1)[j] = \begin{cases} 1 & \text{if } V_0[j] + V_1[j] > t \\ 0 & \text{otherwise.} \end{cases}$$

Definition 1. *A distributed algorithm for the sparse vector sum problem is* privacy-preserving *if there exist probabilistic polynomial-time algorithms M_0, M_1 such that, for all (V_0, V_1),*

$$\{M_0(V_0, SVS(V_0, V_1))\}_{(V_0,V_1)} \overset{\mathbf{c}}{\equiv} \{\mathbf{view}_0(V_0, V_1)\}_{(V_0,V_1)},$$
$$\{M_1(V_1, SVS(V_0, V_1))\}_{(V_0,V_1)} \overset{\mathbf{c}}{\equiv} \{\mathbf{view}_1(V_0, V_1)\}_{(V_0,V_1)},$$

where $\mathbf{view}_0(V_0, V_1)$ (resp., $\mathbf{view}_1(V_0, V_1)$) denotes the view of B_0 (resp., B_1) when the input vectors are V_0 and V_1, and $\overset{\mathbf{c}}{\equiv}$ denotes computational indistinguishability *of* probability ensembles *(See, again, [7] for the definitions of probability ensembles and computational indistinguishability.) The algorithms M_0 and M_1 are called* simulators *(for B_0 and B_1, respectively).*

2.2 ElGamal Encryption

The ElGamal encryption scheme consists of three algorithms, for initialization, encryption, and decryption, respectively.

Initialization The initialization takes a security parameter s as input and outputs an s-bit prime p, another prime q such that $p = 2q + 1$, a cyclic subgroup G of $\mathbf{Z}_p^*$ such that $|G| = q$, a generator g of G, and a pair of keys (x, y) such that $x \in \{0, 1, \ldots, q - 1\}$ and $y = g^x \in G$. Here x is the private key and y is the public key.

Encryption Given the public key y, an encryption of cleartext m is

$$C = E_y(m, r) = (m \cdot y^r, g^r),$$

where r is picked at uniformly at random from $\{0, 1, \ldots, q-1\}$.

Decryption Suppose that $C = (C_1, C_2)$ is a valid ciphertext. Then C can be decrypted using the private key x:

$$m = D_x(C) = \frac{C_1}{C_2^x}.$$

An interesting property of the ElGamal encryption scheme is that it is multiplicatively homomorphic.

Homomorphic Property Let us define the multiplication of two pairs as the multiplication of the corresponding components. Then, we have

$$E_y(m_1 \cdot m_2, r_1 + r_2) = E_y(m_1, r_1) E_y(m_2, r_2).$$

We can similarly define the division operation of pairs. Clearly, ElGamal is also homomorphic with respect to divisions.

$$E_y(m_1/m_2, r_1 - r_2) = E_y(m_1, r_1)/E_y(m_2, r_2).$$

3 Our Algorithm

Without loss of generality, hereafter we assume that all $V_i[j]$ satisfy that $0 \leq V_i[j] \leq t$.

Let n' be a well-known upper bound for the number of customers to which the banks have loaned some money. (In practice, n' can be, for example, the total population of the neighborhood. Note that $n' \ll n$.) Suppose that y is bank B_0's public key and x is the corresponding private key. Our algorithm consists of three stages.

Stage 1: For $\ell = 1, 2, \ldots, n'$, bank B_0 computes

$$J_\ell = E_y(g^{j_\ell}, r_{\ell,0}),$$

where

$$j_\ell = \begin{cases} \text{the } \ell\text{th index } j \text{ s.t. } V_0[j] \neq 0 & \text{if } \ell \leq |\{j : V_0[j] \neq 0\}| \\ n' + 1 & \text{otherwise,} \end{cases}$$

and each $r_{\ell,0}$ is picked uniformly and independently from $\{0, 1, \ldots, q-1\}$.

For $\ell = 1, 2, \ldots, n'$, $k = 1, 2, \ldots, t$, bank B_0 computes

$$U_{\ell,k} = E_y(u_{\ell,k}, r_{\ell,k}),$$

where

$$u_{\ell,k} = \begin{cases} 1 & \text{if } \ell \leq |\{j : V_0[j] \neq 0\}| \text{ and } k \leq V_0[j_\ell] \\ 2 & \text{otherwise,} \end{cases}$$

and each $r_{\ell,k}$ is picked uniformly and independently from $\{0, 1, \ldots, q-1\}$.

Bank B_0 chooses a random permutation σ on $\{1, \ldots, n'\}$, and computes, for each ℓ and each k,

$$J'_\ell = J_{\sigma(\ell)};$$

$$U'_{\ell,k} = U_{\sigma(\ell),k}.$$

Bank B_0 sends $\{J'_\ell\}_{\ell=1,\ldots,n'}$, $\{U'_{\ell,k}\}_{\ell=1,\ldots,n',k=1,\ldots,t}$ to bank B_1.

Stage 2: For $\ell' = 1, 2, \ldots, n'$, bank B_1 computes

$$H_{\ell'} = E_y(g^{h_{\ell'}}, r'_{\ell',0}),$$

where

$$h_{\ell'} = \begin{cases} \text{the } \ell'\text{th index } h \text{ s.t. } V_1[h] \neq 0 & \text{if } \ell' \leq |\{h : V_1[h] \neq 0\}| \\ n' + 2 & \text{otherwise,} \end{cases}$$

and each $r'_{\ell',0}$ is picked uniformly and independently from $\{0, 1, \ldots, q-1\}$.

For $\ell = 1, 2, \ldots, n'$, $\ell' = 1, 2, \ldots, n'$, bank B_1 computes

$$W_{\ell,\ell'} = \begin{cases} (\frac{J'_\ell}{H_{\ell'}})^{\alpha_{\ell,\ell'}} (U'_{\ell,t+1-V_1[h_{\ell'}]})^{\beta_{\ell,\ell'}} & \text{if } \ell' \leq |\{h : V_1[h] \neq 0\}| \\ E_y(X_{\ell,\ell'}, \alpha_{\ell,\ell'}) & \text{otherwise,} \end{cases}$$

where each $\alpha_{\ell,\ell'}$ or $\beta_{\ell,\ell'}$ is picked uniformly and independently from $\{0, 1, \ldots, q-1\}$ and each $X_{\ell,\ell'}$ is picked uniformly and independently from G.

Bank B_1 chooses a random permutation π on $\{1, \ldots, n'\}$, and computes, for each ℓ and each ℓ',

$$W'_{\ell,\ell'} = W_{\ell,\pi(\ell')}.$$

Bank B_1 sends $\{W'_{\ell,\ell'}\}_{\ell=1,\ldots,n',\ell'=1,\ldots,n'}$ to bank B_0.

Stage 3: For $\ell = 1, 2, \ldots, n'$, $\ell' = 1, 2, \ldots, n'$, bank B_0 computes

$$w'_{\ell,\ell'} = D_x(W'_{\ell,\ell'}).$$

Finally, bank B_0 computes a vector O as follows: For each ℓ such that there exists ℓ' such that $w'_{\ell,\ell'} = 1$, bank B_0 sets $O[j_{\sigma(\ell)}] = 1$. All the remaining entries of O are equal to 0. This is defined as the output of our algorithm. Bank B_0 sends this output O to bank B_1. (Note that the n-dimensional sparse vector O should be represented in a compressed form. For example, we can represent it using a list of $(j, O[j])$ such that $O[j] \neq 0$. This will help us achieve low overheads in computation and in communications.)

4 Algorithm Analysis

In this section, we present analysis of the correctness, efficiency, and privacy guarantee of our algorithm.

Correctness Analysis We can show that our algorithm is correct with *high probability* (see [6] for the formal definition of high probability).

Theorem 2. *(Correctness) The output vector O is equal to $SVS(V_0, V_1)$, with* high probability*.*

Proof. We essentially need to show that, for each j, $O[j] = SVS(V_0, V_1)[j]$. We distinguish two cases.

Case 1: $SVS(V_0, V_1)[j] = 1$. In this case, $V_0[j] + V_1[j] > t$. Since $V_1[j] \leq t$, we have $V_0[j] > 0$. Assume j is the ℓth index such that $V_0[j] \neq 0$. Then $j_\ell = j$. For all $k \leq V_0[j]$, we have $u_{\ell,k} = 1$.

Since $V_0[j] \leq t$, we have $V_1[j] > 0$. Assume j is the ℓ'th index such that $V_1[j] \neq 0$. Then $h_{\ell'} = j$. Therefore,

$$\begin{aligned}
W_{\sigma^{-1}(\ell),\ell'} &= \left(\frac{J'_{\sigma^{-1}(\ell)}}{H_{\ell'}}\right)^{\alpha_{\sigma^{-1}(\ell),\ell'}} (U'_{\sigma^{-1}(\ell),t+1-V_1[h_{\ell'}]})^{\beta_{\sigma^{-1}(\ell),\ell'}} \\
&= \left(\frac{E_y(g^{j_\ell}, r_{\ell,0})}{E_y(g^{h_{\ell'}}, r'_{\ell',0})}\right)^{\alpha_{\sigma^{-1}(\ell),\ell'}} (U'_{\sigma^{-1}(\ell),t+1-V_1[h_{\ell'}]})^{\beta_{\sigma^{-1}(\ell),\ell'}} \\
&= (E_y(g^{j_\ell - h_{\ell'}}, r_{\ell,0} - r'_{\ell',0}))^{\alpha_{\sigma^{-1}(\ell),\ell'}} (U'_{\sigma^{-1}(\ell),t+1-V_1[h_{\ell'}]})^{\beta_{\sigma^{-1}(\ell),\ell'}} \\
&= E_y(1, (r_{\ell,0} - r'_{\ell',0})\alpha_{\sigma^{-1}(\ell),\ell'}) \cdot (U'_{\sigma^{-1}(\ell),t+1-V_1[h_{\ell'}]})^{\beta_{\sigma^{-1}(\ell),\ell'}} \\
&= E_y(1, (r_{\ell,0} - r'_{\ell',0})\alpha_{\sigma^{-1}(\ell),\ell'}) \cdot E_y(u_{\ell,t+1-V_1[h_{\ell'}]}, r_{\ell,t+1-V_1[h_{\ell'}]})^{\beta_{\sigma^{-1}(\ell),\ell'}} \\
&= E_y(1, (r_{\ell,0} - r'_{\ell',0})\alpha_{\sigma^{-1}(\ell),\ell'}) \cdot E_y(1, r_{\ell,(t+1-V_1[h_{\ell'}])}\beta_{\sigma^{-1}(\ell),\ell'}) \\
&= E_y(1, (r_{\ell,0} - r'_{\ell',0})\alpha_{\sigma^{-1}(\ell),\ell'} + r_{\ell,(t+1-V_1[h_{\ell'}])}\beta_{\sigma^{-1}(\ell),\ell'}).
\end{aligned}$$

(In the above, the sixth equality is because $t + 1 - V_1[h_{\ell'}] = t + 1 - V_1[j] \leq V_0[j]$.) Consequently, we have $w'_{\sigma^{-1}(\ell),\pi^{-1}(\ell')} = D_x(W_{\sigma^{-1}(\ell),\ell'}) = 1$, which implies that $O[j] = O[j_\ell] = 1$.

Case 2: $SVS(V_0, V_1)[j] = 0$. In this case, $V_0[j] + V_1[j] \leq t$. If $V_0[j] = 0$, then clearly we have $O[j] = 0$. So we only need to consider the case in which $V_0[j] > 0$. Assume j is the ℓth index such that $V_0[j] \neq 0$. Then $j_\ell = j$. For all $k > V_0[j]$, we have $u_{\ell,k} = 2$. For all ℓ', similar to Case 1 we can get

$$W_{\sigma^{-1}(\ell),\ell'} = (E_y(g^{j_\ell - h_{\ell'}}, r_{\ell,0} - r'_{\ell',0}))^{\alpha_{\sigma^{-1}(\ell),\ell'}} E_y(u_{\ell,t+1-V_1[h_{\ell'}]}, r_{\ell,t+1-V_1[h_{\ell'}]})^{\beta_{\sigma^{-1}(\ell),\ell'}}.$$

This means

$$\begin{aligned}
w'_{\sigma^{-1}(\ell),\pi^{-1}(\ell')} &= D_x(W_{\sigma^{-1}(\ell),\ell'}) \\
&= g^{(j_\ell - h_{\ell'})\alpha_{\sigma^{-1}(\ell),\ell'}} \cdot u_{\ell,t+1-V_1[h_{\ell'}]}^{\beta_{\sigma^{-1}(\ell),\ell'}}.
\end{aligned}$$

When $j_\ell \neq h_{\ell'}$, $w'_{\sigma^{-1}(\ell),\pi^{-1}(\ell')}$ is uniformly distributed in G and thus is not equal to 1 with high probability.

When $j_\ell = h_{\ell'}(= j)$, $u_{\ell,t+1-V_1[h_{\ell'}]} = 2$ since $t+1-V_1[h_{\ell'}] = t+1-V_1[j] > V_0[j]$; thus $w'_{\sigma^{-1}(\ell),\pi^{-1}(\ell')}$ is still uniformly distributed in G, being not equal to 1 with high probability.

So, with high probability, we have $O[j] = 0$. ∎

Efficiency Analysis For bank B_0, the main computational overhead is $n'(t+1)$ ElGamal encryptions and $(n')^2$ ElGamal decryptions. For bank B_1, *in the worst case*, the main computational overhead is $2(n')^2$ exponentiations of ElGamal ciphertexts (which is equivalent to $4(n')^2$ exponentiations in G) and n' ElGamal encryptions.

The overall communication overhead is at most $sn'(n'+t+2)$ bits.

Privacy Analysis Finally, we give a brief proof for our privacy guarantee.

Theorem 3. *Our algorithm is privacy-preserving.*

Proof. We first construct simulator M_0 as follows. On input $(V_0, SVS(V_0, V_1))$, M_0 simulates the coin flips of B_0 as described in the algorithm. Then M_0 computes $n^* = |j : SVS(V_0, V_1)[j] > t|$ and randomly chooses a subset L^* of $\{1, \dots, n'\}$ such that $|L^*| = n^*$. M_0 also chooses $\ell'_1, \dots, \ell'_{n^*}$ from $\{1, \dots, n'\}$ uniformly and independently. To simulate each $W'_{\ell,\ell'}$, M_0 uses a random encryption of 1 if $\ell \in L^*$ and $\ell' = \ell'_\ell$; M_0 uses a random encryption of a random cleartext otherwise.

We construct simulator M_1 as follows. On input $(V_1, SVS(V_0, V_1))$, M_1 simulates the coin flips of B_1 as described in the algorithm. Then M_1 simulates $\{J'_\ell\}_{\ell=1,\dots,n'}$, $\{U'_{\ell,k}\}_{\ell=1,\dots,n',k=1,\dots,t}$ using $n'(t+1)$ random encryptions of random cleartexts.

The computational indistinguishability straightforwardly follows from the semantic security of ElGamal. ∎

References

[1] M. Ben-Or, S. Goldwasser, and A. Widgerson. Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. In Proceedings of the 20th Annual ACM Symposium on the Theory of Computing, pp. 1-10, 1988.

[2] D. Chaum, C. Crepeau, and I. Damgard. Multiparty Unconditionally Secure Protocols. In Proceedings of the 20th Annual ACM Symposium on the Theory of Computing, pp. 11-19, 1988.

[3] R. Cramer and I. Damgard. Secure Distributed Linear Algebra in a Constant Number of Rounds. In Advances in Cryptology - Proceedings of CRYPTO 2001, volume 2139 of Lecture Notes in Computer Science, pp. 119-136, 2001.

[4] Michael Freedman, Kobi Nissim, and Benny Pinkas. Efficient Private Matching and Set Intersection. In Proceedings of Eurocrypt 2004, pp. 1-19, May 2004.

[5] L. Kissner and D. Song. Privacy-preserving Set Operations. In Proceedings of Crypto 2005, pp. 241-257, 2005.

[6] O. Goldreich. Foundations of Cryptography, Volume 1. Cambridge University Press, 2001.

[7] O. Goldreich. Foundations of Cryptography, Volume 2. Cambridge University Press, 2003.

[8] O. Goldreich, S. Micali, and A. Wigderson. How to Play ANY Mental Game. In Proceedings of the 19th Annual ACM Symposium on the Theory of Computing, pp. 218-229, 1987.

[9] A. C. Yao. Protocols for Secure Computations. Proceedings of the 21st Annual IEEE Symposium on the Foundations of Computer Science, pp. 160-164, 1982.

[10] A. C. Yao. How to generate and exchange secrets. In Proceedings of the 27th IEEE Symposium on Foundations of Computer Science, pp. 162-167, 1986.

Computer Science and Engineering Department
State University of New York at Buffalo
Amherst, NY 14260, U. S. A.
Email: szhong@cse.buffalo.edu