CERIAS Tech Report 2003-02

# A SECURE PROTOCOL FOR COMPUTING DOT-PRODUCTS IN CLUSTERED AND DISTRIBUTED ENVIRONMENTS

by Ioannis Ioannidis, Ananth Grama, and
Mikhail Atallah

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907

# A Secure Protocol for Computing Dot-Products in Clustered and Distributed Environments

Ioannis Ioannidis, Ananth Grama, and Mikhail Atallah
Department of Computer Sciences, Purdue University,
W. Lafayette, IN 47907.
{ioannis, ayg, mja}@cs.purdue.edu [*]

## Abstract

*Dot-products form the basis of various applications ranging from scientific computations to commercial applications in data mining and transaction processing. Typical scientific computations utilizing sparse iterative solvers use repeated matrix-vector products. These can be viewed as dot-products of sparse vectors. In database applications, dot-products take the form of counting operations. With widespread use of clustered and distributed platforms, these operations are increasingly being performed across networked hosts. Traditional APIs for messaging are susceptible to sniffing, and the data being transferred between hosts is often enough to compromise the entire computation. For example, in a domain decomposition based sparse solver, the entire solution can often be reconstructed easily from boundary values that are communicated on the net. In yet other applications, dot-products may be performed across two hosts that do not want to disclose their vectors, yet, they need to compute the dot-product. In each of these cases, there is a need for secure and anonymous dot-product protocols. Due to the large computational requirements of underlying applications, it is highly desirable that secure protocols add minimal overhead to the original algorithm. Finally, by its very nature, dot-products leak limited amounts of information – one of the parties can detect an entry of the other party's vector by simply probing it with a vector with a 1 in a particular location and zeros elsewhere. Given all of these constraints, traditional cryptographic protocols are generally unsuitable due to their significant computational and communication overheads. In this paper, we present an extremely efficient and sufficiently secure protocol for computing the dot-product of two vectors using linear algebraic techniques. Using analytical as well as experimental results, we demonstrate superior performance in terms of computational overhead, numerical stability, and security. We show that the overhead of a two-party dot-product computation using MPI as the messaging API across two high-end workstations connected via a Gigabit ethernet approaches multiple 4.69 over an un-secured dot-product. We also show that the average relative error in dot-products across a large number of random (normalized) vectors was roughly* $4.5 \times 10^{\Gamma 9}$.

**Keywords:** Secure dot-products, Application-level security protocols, Distributed secure computations.

## 1 Introduction and Motivation

With the emergence of clustered and distributed platforms as cost-effective parallel programming environments, a number of critical computations are routinely executed over the net. These include scientific simulations, data mining operations, and commercial transactions. Data transferred over the net in such applications is highly susceptible to packet sniffing, and other attacks that compromise the privacy and integrity of the computation. While the problem of packet sniffing can be addressed by secure-tunneling the data (encrypting all communication), the overhead of secure tunneling puts tremendous pressure on already scarce communication throughput. The problem of securing communicated data is much better handled in application level protocols as opposed to general cryptographic protocols.

In yet other computations, entities participating in a parallel program may not trust each other. For example, two competing grocery chains may elect to join forces in mining their sales data. Yet, neither might want to reveal precise sales figures to the other. Most database operations take the form of calculating the dot-product of an external with a resident, database vector. These

[*]This work is supported in part by National Science Foundation grants EIA-9806741, ACI-9875899, and ACI-9872101, and by CERIAS at Purdue University. Computing equipment used for this work was supported by National Science Foundation and by the Intel Corp.

vectors are valuable to both parties and leaking information has a definite cost. Nevertheless, the popularity of data mining forces a compromise between speed and security. In more malicious environments, two hosts may choose to execute an untrusted computation protocol to guard against the possibility that one of the hosts may have been compromised. In such cases, hosts may agree to participate in a computation so long as their own data is not compromised (revealed) to others.

In addition to lack of trust in the network and other parties, secure and anonymous protocols are subject to traditional constraints of parallel processing – namely that a ten-fold speedup from parallelism must not be cancelled out by a ten-fold slowdown resulting from the secure protocol. With these objectives in mind, we propose, in this paper, a secure protocol for a key computational kernel – dot-products. By its very nature, a dot-product leaks a certain amount of information. If two parties are trying to compute a secure and anonymous dot-product, one of the parties can detect an entry of the other party's vector by simply probing it with a one in the corresponding location. Due to this, we expect a higher-level application protocol to disallow a large number of probes on a single vector. We focus here on securing a single dot-product using a protocol that has low computational (and communication) overhead, good security properties, and excellent numerical stability.

Consider two parties, Alice and Bob, holding a vector each and wanting to compute securely their dot-product. There are two important aspects of this operation. First, no one eavesdropping the communication should be able to derive any information on the nature of the two vectors. Second, the only thing revealed to either party should be the outcome of the operation. Existing protocols meeting the above criteria derive their security from the properties of modular exponentiation and introduce unacceptable overhead unless perfect security is absolutely necessary. On the other hand there is a wide range of environments demanding a fast protocol, secure enough to resist a large repertory of attacks but not necessarily a very powerful, yet unlikely, adversary. For such applications, linear algebraic techniques provide a powerful set of tools for securing, at the application level, the underlying computations. Here, we present one such protocol that has extremely low overhead (shown to tend to a factor of 4.69 on average), excellent numerical stability, and security properties.

The rest of this paper is organized as follows: Section 2 presents three applications of the proposed technique, Section 3 presents an overview of the protocol and discusses existing results in the area, Section 4 outlines the new dot-product protocol, Section 5 provides experimental results of overhead and stability, and Section 6 draws conclusions and outlines ongoing work.

## 2 Applications

The proposed protocol has been designed with the following specific applications in mind. This list is merely representative of the rich class of applications to which the protocol can be applied.

- *Biometrics and matching*. The heart of all data matching applications is the dot-product. The need for security is heightened when the data contains sensitive information. The most notable example is that of biological or genetic information. The demand for such applications is expected to increase significantly and with it, the need for privacy and efficiency. Anonymity is the main objective but the amount of information involved nullifies any computation-intensive attack, even if it looks feasible in theory.

- *Distributed data mining*. Data mining involves data whose collection has a considerable cost and whose security is essential. The widespread use of data mining techniques and their distributed nature demand fast, secure dot-product protocols. Computation overhead might not be the critical factor in this case, but the limited bandwidth does not allow for a large communication overhead. Any protocol aimed at data mining applications should feature a small number of rounds and a limited bit count.

- *Critical scientific computations*. Such computations usually take place in tightly coupled networks. A typical representative kernel of these applications is an iterative linear system solver. The core of this solver is the sparse mat-vec operation. Matrices are typically row partitioned (typically using a graph partitioner such as Metis or Chaco) and, in this form they are simply multiple dot-products. A defining characteristic of such applications is that data is typically reused over repeated mat-vecs. A protocol exploiting this characteristic, enjoys a huge advantage, because it amortizes most of the overhead over a large number of rounds.

## 3 Overview of Protocol and Related Results

The most commonly used technique for hiding a number is modular multiplication. This is considered a perfectly secure and efficient operation. Generalizing to higher dimensions is what one would ideally want for a protocol involving vectors. In the case of the dot-product, this is not possible due to the incompatibility of the operation with modular arithmetic. Furthermore, in this protocol a vector would have to be multiplied with a random matrix, which leads to complications. The protocol we present is based on the above principle. It leaks a controlled amount of information, however, this information leakage is negligible compared to the inherent leak in the problem. In experiments our protocol demonstrates ideal numerical stability and an overhead, compared to the simple dot-product operation, of a factor of less than 4.7, in the worst case. If the same vector is reused, subsequent computations can reuse more than half of the computations and exactly half of the communication, for an even lower amortized overhead. These measurements were taken over a Gigabit ethernet and they reveal that the overhead is dominated by the communication. Over a slower network, such as the Internet, the above ratio would be even lower.

### 3.1 Related research

The first secure multiparty computation problem was described in [1]. Since then, research on such problems has grown significantly to form one of the most active areas of research in computer security. In [2], a general framework for any secure multiparty computation was presented. Although this framework can be applied to the dot-product problem, yielding an algorithm linear in the size of the vectors, its extensive use of *oblivious transfer* ([5]) makes it impractical because of very high constants in computation and communication.

There have not been many protocols specific to the secure multiparty dot-product computations. In [3, 4], such protocols are described in the context of larger constructions. They are based on conventional cryptographic techniques which incur a large overhead since they rely on the difficulty of problems related to modular exponentiation for their security.

## 4 The Proposed Dot-Product Protocol

We consider here, a two-party computation where each party is *honest-but curious*. In other words, everyone participating in the protocol abides by the rules but if they have the opportunity to find out something more than they are supposed to, they will. We also assume that only one of the parties is interested in the result, a realistic assumption for the range of applications dot-products are associated with. Finally, a random number in the context of real-valued vectors is a uniformly distributed, random integer cast into real.

**Figure 1. Proposed protocol for secure dot-products.**

A formal statement of the problem is as follows: Alice holds vector $v$ and Bob vector $w$, both in $\Re^{d\times 1}$, $d \geq 3$. Alice is interested in computing $w^T \cdot v$ without revealing anything about her vector. Bob is willing to participate in such a protocol as long as he does not reveal more than the dot-product to Alice.

Let $Q$ be a random $s \times s$ matrix, $s \geq 2$, where $s$ is a security parameter. Bob chooses a random $r, 1 \leq r \leq s$, and $s-1$ random $d \times 1$ vectors, $x_i, 1 \leq i \leq s, i \neq r$. Let $w'$ be the vector in $\Re^d$ for which $w_i' = w_i$, for all $1 \leq i \leq d-1$, and $(w_d)' = 1$. He sets $x_r = w'$ and creates an $s \times d$ matrix $X$, whose $i$-th row is $x_i^T$. We define the following:

$$b = \sum_{i=1}^{s} Q_{ir}$$

$$c = \sum_{i=1, i\neq r}^{s} (x_i^T \cdot \sum_{j=1}^{s} Q_{ji})$$

Bob chooses a random $d \times 1$ vector $f$ and three random numbers $R_1, R_2, R_3$, and makes the following public:

$$Q \cdot X$$

$$c' = c + f^T \cdot R_1 \cdot R_2$$

$$g = f \cdot R_1 \cdot R_3$$

Let $v'$ be in $\Re^d$ such that $v_i' = v_i$, for all $1 \leq i \leq d-1$, and $v_d = \alpha$, where $\alpha$ is a random number. Alice computes:

$$y = Q \cdot X \cdot v'$$

$$z = \sum_{i=1}^{s} y_i$$

and

$$a = z - c' \cdot v'.$$

She sends $a$ to Bob and also computes and sends to Bob:

$$h = g^T \cdot v'.$$

Bob computes:

$$\beta = \frac{a + h \cdot \frac{R_2}{R_3}}{b}$$

and sends it to Alice. The desired dot-product is finally given by $\beta - \alpha$. This protocol is illustrated in Figure 1.

## 4.1 Proof of Correctness

A proof that what Bob computes is indeed the desired dot-product is as follows:

$$\begin{aligned} c \cdot v' &= \sum_{i=1; i \neq r}^{s} (x_i^T \cdot \sum_{j=1}^{s} Q_{ji}) \cdot v' \\ &= \sum_{i=1; i \neq r}^{s} (x_i^T \cdot v' \cdot \sum_{j=1}^{s} Q_{ji}) \end{aligned}$$

We also, have:

$$c' \cdot v' = c \cdot v' + h \cdot \frac{R_2}{R_3}$$

$$\Rightarrow a + h \cdot \frac{R_2}{R_3} = z - c \cdot v'$$

$$y = Q \cdot X \cdot v' = Q \cdot (X \cdot v') = Q \cdot u$$

Where,

$$u_i = x_i^T \cdot v'$$

From this, we have

$$\begin{aligned} y_j &= \sum_{i=1}^{d} (Q_{ji} \cdot x_i^T \cdot v') \\ \Rightarrow z &= \sum_{j=1}^{s} y_j \\ &= \sum_{j=1}^{s} \sum_{i=1}^{d} (Q_{ji} \cdot x_i^T \cdot v') \\ &= \sum_{i=1}^{d} (x_i^T \cdot v' \cdot \sum_{j=1}^{s} Q_{ji}) \\ \Rightarrow z - c \cdot v' &= x_r^T \cdot v' \cdot \sum_{j=1}^{s} Q_{jr} = b \cdot (w')^T \cdot v' \end{aligned}$$

Therefore, $\beta = w^T \cdot v + \alpha$. This verifies the correctness of the protocol.

## 4.2 Analysis of Security

Due to the nature of multiplication and addition, some information is statistically revealed by the protocol. However, since exposing a vector $d$ times in dot-product calculations will reveal it fully, any such analysis can only have limited success. We analyze here the information flow between Alice and Bob.

- From Bob to Alice

  Bob makes three things public:

  - $Q \cdot X$. Vector $w$ is involved in this $s \times d$ matrix. However, even if Alice can guess which row of $X$ is $w$, it is impossible to separate the two matrices in any way without further information. Therefore, making this matrix public conveys no information about $w$ to Alice.
  - $c'$. This is the sum of a random vector and a vector, which could be claimed to leak information, although it is quite debatable whether it does. However, unless the random vector is disclosed, $c'$ is certain not to leak information.
  - $g$. This vector does not reveal information about the random vector of the previous paragraph, unless one could guess the $R_i$'s. So long as they are random, no information is leaked.
  - $a, h$. Alice knows that the product is a linear combination of these two. If she uses a linearly independent vector $w''$ w.r.t. $w$, to create more equations so that she can deduce the coefficients for $a$ and $h$, she will introduce a new unknown to the system for each new equation, namely $w''^T \cdot v$. If $w''$ is linearly dependent w.r.t. $w$ she does not create a new equation. We, also, note that $a = 0$ if and only if $h = 0$. Therefore, she cannot deduce any information about $v$.

    Sending $h$ to Bob, Alice reveals an equation about her vector. This way, both parties come out of the protocol with the same number of equations. To keep this equation secret from an eavesdropper, Alice and Bob must share a secret random number or use standard encryption to communicate it. Since this is only a scalar, the overhead is insignificant.

- From Alice to Bob.

  Alice sends two things to Bob, after step 1.

– $a$. This vector cannot reveal anything to Bob, unless he can separate $z$ from $c^0 \cdot v$.

– $h$. This gives to Bob an equation about $v^0$. Computing the dot-product gives another equation. He cannot combine $a$ and $h$ to get any more information for $w$. Totally, Bob has two equations about $v^0$. However, for $v$ itself, he effectively has one equation, as many as Alice has for his vector.

### 4.3 Stability considerations

Besides the security and efficiency of the scheme, we need to ensure that the error introduced by the extra computations does not affect the accuracy of the result. One can avoid such errors by extending the precision, however, this introduces extra overhead. We present experimental data suggesting that the accuracy of the computation is similar to that of computing an un-secured dot-product without extended precision.

## 5 Experimental results

As we have indicated earlier, we are interested in two aspects of the protocol behavior: the overhead it introduces and its numerical stability.

- Overhead. There are two types of overhead involved in the protocol. The communication overhead is the amount of extra data that needs to be communicated, compared to the simple protocol. The computation overhead refers to the extra computation performed. We note that both of these overheads depend on the security parameter $s$. The computation overhead has a quadratic dependence on it, while the communication overhead is only linearly dependent. All measurements presented in this section correspond to $s = 2$. We believe that even for this low value of the security parameter, there is a sufficient measure of security provided by the protocol.

  – Communication. Bob must send four vectors to Alice. Two further rounds of communication are needed, involving in total three scalars.

  – Computation. One of the features of our protocol is that a large part of the data can be reused when the same vector is repeatedly used in different dot-product computations. More specifically, if Alice and Bob have engaged in an execution of the protocol and they want to execute the protocol again with only Alice changing her vector, Bob needs to change only $f$, $R_1$, $R_2$, $R_3$ and, consequently, $c^0$ and $g$. $Q \cdot X$ and $b$ can remain the same without compromising the security of his vector. This means that the majority of the computation overhead can be amortized over a large number of executions.

  – Total overhead. The protocol was implemented on two PIII/450MHz machines connected over a Gigabit ethernet. The messaging API was MPI. All the measurements were taken over the entire protocol, without any amortization. We used randomly instantiated, normalized vectors of length $10^6$. For the generation of random numbers, we used the standard C-library drand48 pseudo-random number generator. The overhead in this case is observed to be $\approx 4.69$, on average. We observe that in spite of the relatively fast network, communication overhead dominates total overhead. One can expect this effect to be even more pronounced for slower networks. The use of more sophisticated means for obtaining random numbers should not have significant impact on the above overhead.

- Numerical stability. The average relative error over a large number of random (normal) vectors was measured to be $4.493 \cdot 10^{\Gamma 9}$. Given that the relative error of the normal computation of the dot-product is of the same order, the results can be considered absolutely satisfactory.

## 6 Concluding Remarks and Ongoing Research

In this paper, we have presented a secure protocol for computing dot-products. We have demonstrated analytically as well as experimentally the excellent performance characteristics, numerical stability, and security properties of the protocol. Compared to conventional cryptographic techniques, this linear algebraic technique has much lower overhead.

Ongoing work in our group focuses on developing secure computation techniques for a variety of problems in data mining and analysis. Data mining poses the problem of computing the dot-product between two binary vectors, efficiently and securely. The challenge in this case is to avoid the extensive use of number theoretic techniques. Two variations of the problem come from the dot-product between sparse vectors (the difficulty arising from the need to hide which positions are important while not wanting to traverse all of them) and the threshold variation of the problem, where we want to know whether the product exceeds a specified threshold but nothing more.

## References

[1] Andrew C.-C. Yao, ”Protocols for secure computation”, Proc. 23rd IEEE Symposium on Foundations of Computer Science (FOCS), 1982, pp 160 - 164.

[2] Oded Goldreich, Silvio Micali and Avi Wigderson, ”How to play any mental game or a completeness theorem for protocols with honest majority”, Proc. 19th Annual ACM Symposium on Theory of Computing (STOC), 1987, pp. 218 - 229.

[3] Wenliang Du and Mikhail J. Atallah, ”Protocols for Secure Remote Database Access with Approximate Matching”, 7th ACM Conference on Computer and Communications Security (ACMCCS 2000), The First Workshop on Security and Privacy in E-Commerce, Nov. 1-4 2000, Athens, Greece.

[4] Wenliang Du and Mikhail J. Atallah, ”Privacy-Preserving Cooperative Scientific Computations”, In 14th IEEE Computer Security Foundations Workshop, June 11-13 2001, Nova Scotia,Canada.

[5] M. Rabin, ”How to exchange secrets by oblivious transfer”, Technical report TR-81, Aiken Computation Laboratory, Harvard University, 1981.