CERIAS Tech Report 2003-39

AN EFFICIENT PROTOCOL FOR YAO'S MILLIONAIRES' PROBLEM

by Ioannis Ioannidis, Ananth Grama

Center for Education and Research in
Information Assurance and Security,
Purdue University, West Lafayette, IN 47907-2086

An Efficient Protocol for Yao's Millionaires' Problem

Ioannis Ioannidis and Ananth Grama

Department of Computer Sciences, Purdue University,
W. Lafayette, IN 47907.
{ioannis, ayg}@cs.purdue.edu *

Abstract

The increase in volume and sensitivity of data communicated and processed over the Internet has been accompanied by a corresponding need for e-commerce techniques in which entities can participate in a secure and anonymous fashion. Even simple arithmetic operations over a set of integers partitioned over a network require sophisticated algorithms. As a part of our earlier work, we have developed a secure protocol for computing dot products of two vectors. In this paper,we present a secure protocol for Yao's millionaires' problem. In this problem, each of the two participating parties have a number and the objective is to determine whose number is larger without disclosing any information about the numbers. This problem has direct applications in on-line bidding and auctions. Furthermore, combined with a secure dot-product, a solution to this secure multiparty computation provides necessary building blocks for such basic operations as frequent item-set generation in association rule mining.

Although an asymptotically optimal solution for the secure multiparty computation of the 'less-or-equal' predicate exists in literature, this protocol is not suited for practical applications. Here, we present a protocol which has a much simpler structure and is more efficient for numbers in ranges practically encountered in typical e-commerce applications. Furthermore, advances in cryptanalysis and the subsequent increase in key lengths for public-key cryptographic systems accentuate the advantage of the proposed protocol. We present experimental evidence demonstrating the efficiency of the proposed protocol both in terms of time and communication overhead.

Keywords: secure multiparty computations, Yao's millionaires' problem, on-line bidding

1 Introduction and Motivation

With e-commerce having overcome the status of an emerging technology and having established itself as an important part of international trade, anonymity and protection of private information have become critical attributes for many applications. Computations involving sensitive data are routinely performed over the net. Databases are accessed for information, commercial transactions are completed without any physical contact, and an increasing number of people demand fast and secure net-based services. While securing the links of a network (using secure tunneling, application level messaging APIs, etc.) is a relatively easy task, security against untrusted network

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

entities and hosts poses considerably greater computational challenges. Elaborate and often computationally expensive cryptographic techniques must be used to prevent leakage of information to such entities and the associated fallout. Problems of this nature are collectively known as *secure multiparty computation* problems.

The first secure multiparty computation problem was described by Yao in [1]. Consider two parties, Alice and Bob, holding numbers a and b, respectively. Yao's millionaires' problem is the problem of computing the predicate $a \geq b$, without disclosing anything more than the result to either party. A problem of historical and theoretical importance for cryptography, it gains new significance in the context of e-commerce and data mining. Commercial applications often have to execute multiple comparisons of numbers. In many cases, these numbers contain sensitive information, which must be secured. To reduce the communication overhead and avoid network bottlenecks, it is desirable that these operations are completed without resorting to a third-party. All of these applications can use the proposed protocol as a kernel operation for secure comparisons. In spite of the relative simplicity of the problem statement, its scope, significance, and complexity of solutions cannot be understated.

The most obvious application of secure comparisons in the context of e-commerce comes from auction pricing. Auction pricing has, at its core, secure comparison of bids. The large throughputs required from computational infrastructure supporting these applications place emphasis on efficiency (both computation and communication) as well as ease of implementation. For a secure comparison protocol to be of practical interest in this setting, it needs to be adapted to the nature of the compared numbers and the state of the underlying network.

A further range of applications comes from data mining. As an example, we note the use of FP-trees ([10]) for frequent set computations. In FP-trees, each node represents a frequent set. To verify whether a set is indeed supported (i.e., is frequent) by the database, a comparison must be performed during each step. An involved cryptographic protocol is less likely to gain wide acceptance and is more susceptible to implementation level security flaws.

In this paper, we present a protocol that is particularly suitable for a variety of practical applications. Its performance on numbers in ranges frequently encountered in Internet applications is significantly better than earlier known results. A frequently overlooked performance factor is the nature of the underlying communication network between participating entities. In conventional wide-area networks, latencies in the range milliseconds often dominate bandwidth terms for short messages. We demonstrate that in such scenarios, our protocol provides significant additional performance gains. We support these assertions with experimental data from an implementation of the protocol across two geographically distributed hosts. Finally, the simplicity of our protocol makes it particularly attractive for implementation at the core of various data mining applications.

The rest of the paper is organized as follows: Section 2 presents an overview of the protocol and discusses existing results in the area, Section 3 outlines the proposed protocol, Section 4 provides experimental data relating to overhead and scalability, and Section 5 draws conclusions and summarizes the results of the paper.

2 Overview of Protocol and Related Results

A comparison of two numbers can be carried out simply by examining the most significant bit in which they differ. Identical bits do not affect the result, while the effect of unequal low-order bits is overshadowed by the high-order bit. Based on this principle, we can outline a protocol for securely comparing two numbers:

Alice creates two numbers for each bit of her number. One of the numbers encodes the result of the operation if this bit is the decisive one, the other is a dummy, having no effect on the outcome. The former must also cancel the effect of any other number corresponding to less significant bits, when combined with them. Bob secretly examines only the numbers he is interested in and combines all the numbers he has seen to derive the result of the comparison. Of course, all the numbers must be encrypted, so that Bob does not get information about individual bits of Alice's number.

Indeed, the above protocol allows Bob to build an encrypted version of the difference of the two numbers. In the final step, Bob is allowed to decrypt the sign of the difference, but nothing else. This encoding is built on

pieces of information communicated from Alice, which individually or partially combined reveal nothing to Bob, but the combination of all of them reveals the desired result. This is reminiscent of the well-known technique of *secret sharing*, where a bit b XORed with a random bit r carries no information until r is revealed.

The time and communication complexity of the protocol are suboptimal. For each bit, a share of the final encoding must be passed to Bob. Such a share cannot be shorter than the numbers themselves, leading to quadratic complexities. The advantage of the scheme comes from the fact that secret sharing is extremely efficient and simple to implement and that Bob can receive all the shares of interest to him in one round. This minimizes the number of communication rounds and associated network latency overhead.

2.1 Related Research

The first protocol for solving the secure comparison problem was presented by Yao [1]. The described solution had exponential time and space requirements but it paved the way for research on multiparty computation problems. In [8], a general framework for any secure multiparty computation problem is presented. This approach to secure function evaluation relies on a circuit constructor A, which scrambles the bits on the wires of the circuit by replacing each with a random token. It then encrypts the truth tables of all gates accordingly so that two tokens together decrypt the corresponding token on the outgoing wire, and provide the interpretation for the tokens appearing in the circuit output. It sends the encrypted circuit to B (the circuit evaluator), who obtains the tokens corresponding to his input bits using one-out-of-two oblivious transfer (described briefly in the next section). This ensures that he learns nothing about other tokens. B is then able to evaluate the circuit and to compute the output on his own. In the case of Yao's millionaires' problem, this framework yields a linear time and communication protocol. The disadvantage of this solution is that a boolean circuit of the comparison operation needs to be built, introducing a large number of oblivious transfers, communication rounds, and associated implementation difficulties.

Beaver et al. [3] give a constant-round cryptographic protocol for multi-party computation. Fair protocols for two-party computation (and extensions to multiple parties) have been investigated by Chaum et al. [6], Beaver and Goldwasser [2], and by Goldwasser and Levin [9]. They combine oblivious circuit evaluation with gradual release techniques to obtain fairness.

Feige et al. [7] study an extension of the multi-party secure computation models using a third party T, which receives a single message, does some computation, and outputs the function value, but does not learn anything else about the inputs. In a related model, Cachin and Camenisch [4, 5] present a scheme in which T is not involved in regular computations and only used in case some party misbehaves. The 1-out-of-n oblivious transfer primitive [11] is a method for allowing a party to access one of n secrets, without this party acquiring any information about the other $n-1$ secrets, and also not disclosing any information about their choice (which of the $n-1$ was selected).

3 Proposed Protocol for Yao's Millionaire's Problem

We begin the formal description of our protocol with a brief overview of *oblivious transfer* [11]. An oblivious transfer of one bit can be described as follows: the sender S has two bits as input, b_0 and b_1. The receiver R chooses to receive b_i from S, $i \in \{0, 1\}$. S sends OT(b_i) to receiver R in such a way that (i) the receiver obtains the value b_i without receiving any information about $b_{\overline{i}}$; and (ii) the sender does not know the value of i. Oblivious transfer forms the basis for many cryptographic protocols. We can now describe our protocol as follows:

We assume that Alice and Bob have already agreed that their numbers are less than 2^d. We denote Alice's number a and Bob's number b.

1. Alice creates a $d \times 2$ matrix A of k-bit numbers, where k is the length of the oblivious transfer key. She also picks random $r, s, 0 \leq r < 2 \cdot k$ and $s \leq k$, with s being *large enough.* We will subsequently formalize the

notion of 'large enough'.

2. Let A_{ijl} denote the l-th bit of A_{ij}, where A_{ij0} is the least significant bit. For every i, $1 \leq i \leq d$, Alice does the following:

   (a) For every $j \geq s$, set A_{i1j} and A_{i2j} to random bits.

   (b) Let $l = 1$, if $a_i = 1$ and $l = 2$, otherwise. For every j, $0 \leq j \leq 2 \cdot i - 1$, set A_{ilj} to a random bit.

   (c) Let $m = 2 \cdot i$. Set $A_{il(m+1)}$ to 1. Set A_{ilm} to a_i.

   (d) For every i, $1 \leq i < d$, create a random k-bit number S_i. Create S_d so that all its bits except the two high-order ones are random. Set

   $$S_{d(k-1)} = 1 \oplus \bigoplus_{j=1}^{d-1} S_{j(k-1)} \oplus \bigoplus_{j=1}^{d} A_{j1(k-1)}$$

   and

   $$S_{d(k-2)} = 1 \oplus \bigoplus_{j=1}^{d-1} S_{j(k-2)} \oplus \bigoplus_{j=1}^{d} A_{j1(k-2)},$$

   where $\bigoplus$ denotes the bitwise XOR operation.

   (e) For $l = 1, 2$, set

   $$A'_{il} = leftrot(A_{il} \oplus S_i, r),$$

   where $leftrot(x, y)$ denotes the operation of rotating to the left number x by y bits.

3. For every i, $1 \leq i \leq d$, Bob obliviously transfers A'_{il}, where $l = b_i + 1$.

4. Alice sends $S = leftrot(\bigoplus_{j=1}^{d} S_j, r)$ to Bob.

5. Bob calculates the bitwise XOR of all the values he received through oblivious transfer and S. He scans the result from left to right until he reaches a *large* (again, we will subsequently formalize the notion of 'large') number of continuous 0's. The bit to the right of the first non-zero bit ending the streak of 0's signifies the result of the comparison. If this bit is 1, $a \geq b$, otherwise, $a < b$.

3.1 Correctness

We prove the correctness of the proposed protocol as follows: observe that Bob derives the result of the comparison from

$$S \oplus \bigoplus_{i=1}^{d} A'_{i(b_i+1)} = leftrot(\bigoplus_{i=1}^{d} A_{i(b_i+1)}, r).$$

Effectively, the result depends on

$$c = \bigoplus_{i=1}^{d} A_{i(b_i+1)}.$$

We note that every entry of A, and therefore c, can be split into two parts. The leftmost part is a random pattern, not affecting the result in any way. The rightmost part carries the necessary information. The two parts are separated by a zone of zeros. The length of each partition of c is linked to the security of the scheme.

For each i, only one of A_{i1} and A_{i2} has a non-zero rightmost part. This is A_{i1}, if $a_i = 1$, otherwise, it is A_{i2}. In addition, if $i > j$, and A_{il} has a non-zero rightmost part, $A_{il} \oplus A_{jm}$, $m = 1, 2$, has a non-zero rightmost part

and the two leftmost bits of that part will be identical to those of A_{il}. The result of the above is that the rightmost part of c is a function of only the entries Bob has chosen to transfer that correspond to unequal bits in a and b and that the only non-random bits of that part are the two leftmost, which encode the result of $a_i > b_i$, where i is the highest-order bit in which a and b differ. Due to the separating zone of zeros, Bob will be able to discern the left edge of the rightmost part of c, even after the rotation. If $a_i > b_i$, the two bits indicating the result will read 11 and Bob will answer that $a \geq b$. If $a_i < b_i$, they will read 10 and the answer will be $a < b$. If $a = b$, there will be no rightmost part in c. However, the two leftmost bits in c will read 11 and will be the ones indicating the result. In this case, Bob will answer that $a \geq b$, which is the correct answer.

3.2 Security

We prove the security of the proposed protocol by examining the information flow between Alice and Bob.

- *From Alice to Bob*

  There are three kinds of numbers that Bob receives from Alice.

  - $leftrot(A_{i(1+b_i)} \oplus S_i, r)$. Since for every i Bob receives only one such number and S_i is random, no information is leaked from these numbers or any combination thereof.
  - S. This is the bitwise XOR of random numbers and it carries no information. It reveals information only if XORed with $\bigoplus_{i=1}^{d} leftrot(A_{i(1+b_i)}, r)$. The result of this operation is c.
  - c. The leftmost part of c is just a random string. The rightmost part is a random string as well, except the two leftmost bits, which reveal the outcome of the operation. The only possible source of leakage of information is the position of the indicating bits. However, to deduce any information from their position, one must either deduce information about the value of r or the length of the separating zone. In the case of r, it is random and the probability of guessing it correctly is at most as large as guessing the decisive position in a and b before executing the protocol. On the other hand, if the length of the separating zone is random and *large*, deriving any information from the position of the indicating bits is as easy as guessing the length of the zone. Allowing the length to be a uniformly distributed random number between $r' \cdot d$ and $(r'+1) \cdot d$, where r' is also a uniformly distributed random number between 1 and d, makes guessing the length of the separating zone as probable as guessing the position of the decisive bit before the execution of the protocol. In either case, the probability distribution remains the same and Bob learns nothing from the communication with Alice, besides the result of the operation.

- *From Bob to Alice*

  Alice can receive information concerning b only through the oblivious transfers. Provided this operation is secure, Alice can learn nothing from the protocol.

3.3 Complexity and efficiency

We now examine the computational and communication complexity of the proposed protocol. Alice must construct a $O(d)$-bit number for each bit of her number. Bob has to XOR $O(d)$ numbers of $O(d)$ length. The time complexity for both parties is $O(d^2)$. Similarly, $O(d)$ numbers of $O(d)$ length each are communicated. The communication complexity of the protocol is $O(d^2)$, as well. Although, asymptotically the protocol is suboptimal both in terms of time and communication, a linear time or communication protocol would have the advantage only for numbers of length larger than the key length. Given that almost all cryptographic libraries work with keys of length at least 512 bits and most recommend keys of 1024 bits, suboptimality is an issue for virtually no application.

Moreover, there are only two rounds in the protocol. In the first all the oblivious transfers are performed. In the second, the result is communicated across. This is the minimal number of rounds one can expect from a security protocol. As noted before, the number of rounds is of primary of concern for any protocol taking place over a high-latency network.

Finally, we note that there are only d 1-out-of-2 oblivious transfers performed. Although for asymptotic analysis, a 1-out-of-O(1) oblivious transfer is a constant time operation, this is an oversimplification. The most common way of implementing 1-out-of-2 oblivious transfer involves four encryptions and two decryptions in a commutative encryption scheme, like RSA. Public-key cryptography is extremely expensive and for all practical purposes, a quadratic number of cheap operations, like bitwise XOR, is preferable to a linear, yet large, number of oblivious transfers. Since it is unlikely that a protocol for Yao's millionaires' problem with less than the equivalent of d 1-out-of-2 oblivious transfers exists, we hypothesize that this is the minimum number of encryption operations one can expect from a protocol for the specific problem.

4 Experimental results

As we mentioned before, most cryptographic libraries work only with 512-bit keys. For the security of the protocol, d^2 must be less than the key length. This allows the comparison of numbers up to ≈ 20 bits long. For applications in e-commerce, where the numbers involved are prices, numbers of length of 20 bits are adequate for any class of applications. We report here measurements for numbers of length 7, 15 and 20 bits. We observe the communication and computation overhead, relative to the simple, unsecured protocol of exchanging a pair of numbers. The protocol was implemented on two PIII/450MHz machines across an ethernet. Oblivious transfer was implemented using the RSA implementation of the Crypto++ 3.2 cryptogrphic library [12].

The results of our implementation are shown in Table 1. The simple communication cost refers to the cost of transferring a single number of specified length across the network. The domination of computation overhead over all communication is evident in the table (in the worst case, 290 ms of computation vs. 4 ms of communication for a 20 bit comparison). This figure would change if the same experiment were to be carried out across geographically dispersed hosts. It is also evident that computation scales linearly with bit length, confirming the significance of minimizing the number of encryption operations, even at the expense of increasing other complexities. Finally, the effect of network latency is evident: there is only a marginal increase in communication time with the increase of bit count.

Length	Communication (simple)	Communication Overhead	Computation Overhead
7	3.1	3.65	101
15	3.1	3.80	217.5
20	3.1	3.98	290

Table 1. Experimental measurements (in msecs) of the protocol for varying bit lengths across two Pentium III/450 MHz machines connected on an Ethernet. The simple communication time refers to the time for exchanging two numbers of specified bit-length.

5 Concluding Remarks

In this paper, we have presented a secure protocol for comparing two numbers. We have demonstrated analytically as well as experimentally the performance characteristics and security of the protocol. Compared to existing

schemes, this protocol is more efficient for practical applications and is easy to implement. We have also shown that the range of numbers for which this protocol is superior to other related protocols will only increase with increase in key-length.

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] D. Beaver and S. Goldwasser, *Multiparty computation with faulty majority,* in Proc. 30th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 468-473, 1989.

[3] D. Beaver, S. Micali, and P. Rogaway, *The round complexity of secure protocols,* in Proc. 22nd Annual ACM Symposium on Theory of Computing (STOC), pp. 503- 513, 1990.

[4] Christian Cachin, *Efficient Private Bidding and Auctions with an Oblivious Third Party*, Proc. 6th ACM Conference on Computer and Communications Security (CCS), 1999. p. 120 - 126

[5] Christian Cachin and Jan Camenisch, *Optimistic Fair Secure Computation*, In Mihir Bellare, editor, Advances in Cryptology: CRYPTO 2000, Vol. 1880, Lecture Notes in Computer Science, pages 94-112. Springer, 2000.

[6] D. Chaum, I. Damgard, and J. van de Graaf, *Multiparty computations ensuring privacy of each party's input and correctness of the result,* in Advances in Cryptology: CRYPTO '87 (C. Pomerance, ed.), Vol. 293, Lecture Notes in Computer Science, Springer, 1988.

[7] U. Feige, J. Kilian, and M. Naor, *A minimal model for secure computation,* in Proc. 26th Annual ACM Symposium on Theory of Computing (STOC), pp. 554-563, 1994.

[8] 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.

[9] S. Goldwasser and L. Levin, *Fair computation of general functions in presence of immoral majority,* in Advances in Cryptology: CRYPTO '90 (A. J. Menezes and S. A. Vanstone, eds.), Vol. 537, Lecture Notes in Computer Science, Springer, 1991.

[10] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 SIGMOD International Conference on Management of Data*, pages 1-12, Dallas, Texas, May 2000.

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

[12] http://www.eskimo.com/ weidai/cryptlib.html