Private Outsourcing of Matrix Multiplication over Closed Semi-rings

Mikhail J. Atallah[1], Keith B Frikken[2] and Shumiao Wang[1]
[1]*Department of Computer Science, Purdue University, West Lafayette, IN, U.S.A.*
[2]*Department of Computer Science and Software Engineering, Miami University, Oxford, OH, U.S.A.*

Keywords: Privacy-preserving Protocols, Private Outsourcing.

Abstract: Many protocols exist for a client to outsource the multiplication of matrices to a remote server without revealing to the server the input matrices or the resulting product, and such that the server does all of the super-linear work whereas the client does only work proportional to the size of the input matrices. These existing techniques hinge on the existence of additive and multiplicative inverses for the familiar matrix multiplication over the $(+,*)$ ring, and they fail when one (or both) of these inverses do not exist, as happens for many practically important algebraic structures (including closed semi-rings) when one or both of the two operations in the matrix multiplication is the "min" or "max" operation. Such matrix multiplications are very common in optimization. We give protocols for the cases of $(+,\min)$ multiplication, $(\min,\max)$ multiplication, and of $(\min,+)$ multiplication; the last two cases are particularly important primitives in many combinatorial optimization problems.

1 INTRODUCTION

In secure computational outsourcing, a client outsources to a server a computationally expensive task, such that the server learns neither the inputs nor the output, yet carries out the bulk of the computational burden whereas the client does an amount of computing that is proportional to the size of the input. For the usual $(+,*)$ matrix multiplication, this means that the server does all the super-linear work (cubic if it uses the most commonly used algorithm for carrying out the product), and the client does work proportional to the number of entries in the input matrices. For the massive data arising in engineering, the physical sciences, and economic and market modeling, matrix multiplication is a notoriously expensive operation and is therefore a prime candidate for computational outsourcing. Even modular exponentiation has been the subject of extensive protocol design for the purpose of getting the remote server(s) to do the cubic amount of work (in the number of bits) with the local client doing only quadratic work (i.e., a constant number of integer multiplications); of the long sequence of papers on this topic, we refer the reader to (Hohenberger and Lysyanskaya, 2005a).

The techniques used in the existing protocols for the secure outsourcing of $(+,*)$ matrix multiplication, do not work in the cases of $(\min,+)$, $(\min,\max)$, and $(+,\min)$ multiplications, because in these cases one or both of the algebraic operations does not have an inverse. This paper presents efficient secure outsourcing protocols for these cases (hence for the cases where the roles of max and min are interchanged). Of particular practical importance are the $(\min,+)$ and $(\min,\max)$ cases. The former arises in many contexts, one of which is the MAP inference problem in graphical models including cyclic and skip-chain models that arise in many applications (Felzenszwalb and McAuley, 2011). It also arises in the context of optimizing a sequence of tasks where the cost of performing a task depends on the state and there is a cost incurred in moving between states (Saks, 1988). It is in fact a standard part of textbooks, usually in the chapters on graph algorithms (see, e.g., Section 25.1 of (Cormen et al., 2001)). The $(\min,\max)$ matrix multiplication (or equivalently, $(\max,\min)$) arises in the context of bottleneck optimization problems, where one wants to optimize a quantity subject to upper limits constraints (i.e., maximum capacity) or, symmetrically, lower limit constraints (see, e.g., (Duan and Pettie, 2009), (Vassilevska et al., 2007) and the bibliographic references therein). The case for outsourcing such computations is compelling not only because matrix multiplication is expensive and the matrices that arise in the application domains are huge, but also because it is rarely the case that a single such multiplication suffices: In the above-mentioned combinatorial optimization problems a (possibly long) sequence of

J. Atallah M., B Frikken K. and Wang S..
Private Outsourcing of Matrix Multiplication over Closed Semi-rings.
DOI: 10.5220/0004054101360144
In *Proceedings of the International Conference on Security and Cryptography* (SECRYPT-2012), pages 136-144
ISBN: 978-989-8565-24-2

$(\min,+)$ or $(\max,\min)$ multiplications is needed to solve the combinatorial optimization problem at hand (Duan and Pettie, 2009; Vassilevska et al., 2007; Cormen et al., 2001). In cases where the input matrices, the intermediate matrices, and the output matrices are confidential (e.g., too revealing of proprietary models or organizational data and operations), this outsourcing needs to be done in the privacy-preserving framework considered in this paper.

This paper is organized as follows. Section 2 gives the problem definition, Section 3 reviews related work. Section 4 gives the protocol for the $(+,\min)$ multiplication, Section 5 for the $(\min,\max)$ multiplication, Section 6 for the $(\min,+)$ multiplication, and Section 7 concludes.

2 PROBLEM DEFINITION

The main goal of this paper is to compute various types of generalized matrix multiplication in a private outsourced manner. Given $N \times N$ matrices A and B where each entry is in the range $[0,\Sigma)$. We consider the following types of computations[1]:

1. $(+,\min)$: That is if $C = AB$, then $C[i,j] = \sum_{\ell=1}^{N} \min\{A[i,\ell], B[\ell,j]\}$.
2. $(\min,\max)$: That is if $C = AB$, then $C[i,j] = \min_{\ell\in[1,N]}\{\max\{A[i,\ell], B[\ell,j]\}\}$.
3. $(\min,+)$: That is if $C = AB$, then $C[i,j] = \min_{\ell\in[1,N]}\{A[i,\ell] + B[\ell,j]\}$.

To achieve these computations we introduce protocols that use arithmetic circuits (i.e., circuits using the operations: addition, multiplication by constant, and multiplication). Our protocols take the following form: the client does some pre-processing to create the inputs to an arithmetic circuit (or circuits), the computation of the arithmetic circuits is privately outsourced to a server (or servers), and the client does some post-processing on the outputs of the circuits to obtain the results. The goals are to minimize the client's work, the server's work, the multiplicative depth of the circuit, and the number of rounds in the protocol. Several techniques exist for the private outsourcing of arithmetic circuits, and thus these techniques allow us to privately outsource the computation in the following scenarios:

1. The Sharemind system(Bogdanov et al., 2008) allows 3 servers to compute a result, and is secure against any single honest-but-curious server. This system requires 2 rounds of interaction for multiplication, but multiplications can be done in parallel. Thus the number of communication rounds is the depth of the circuit, and the total computation is the size of the circuit. Furthermore, the client's computation is proportional to the number of inputs in the circuit. Another version of this, is to use the generic results in SMC (such as (Goldreich et al., 1987)), which would allow n servers where more than two-thirds of the servers are honest, but the dishonest servers are actively corrupted.
2. There are many somewhat homomorphic encryption schemes, that allow a bounded multiplicative depth circuit to be computed. For example, the scheme in (Boneh et al., 2005) allow a single multiplication to be performed. Other schemes (Naehrig et al., 2011; Brakerski and Vaikuntanathan, 2011b; Brakerski and Vaikuntanathan, 2011a) support a larger multiplicative depth, but the schemes are more efficient when the depth is small. These schemes allow a client to outsource the computation to a single server by doing work proportional to the number of inputs to the circuit, and the server does work proportional to the size of the circuit.

Thus by providing schemes that use circuits of limited multiplicative depth to compute the desired matrix multiplications, we provide a mechanism to securely compute the matrix multiplications. That is, all that the server (or servers) see if the results that they see in the arithmetic circuit computation, and the computations have the same structure (regardless of the input). Hence, if the underlying protocol is secure then the resulting protocol based on these computations will also be secure.

Table 1 summarizes the results presented in this paper where Client denotes the amount of computations performed by the client, Server denotes the amount of computation performed by the server, dep is the multiplicative depth of the computations, and rds is the number of rounds of interaction that are required.

Table 1: Result Summary.

Operation	Client	Server	Dep	Rds
$(+,\min)$	$O(\lvert\Sigma\rvert N^2)$	$O(\lvert\Sigma\rvert N^3)$	1	1
$(+,\min)$	$O(\sqrt{\lvert\Sigma\rvert} N^2)$	$O(\sqrt{\lvert\Sigma\rvert} N^3)$	2	1
$(\min,\max)$	$O(\lvert\Sigma\rvert N^2)$	$O(\lvert\Sigma\rvert N^3)$	1	1
$(\min,\max)$	$O(\sqrt{\lvert\Sigma\rvert} N^2)$	$O(\sqrt{\lvert\Sigma\rvert} N^3)$	3	2
$(\min,+)$	$O(\lvert\Sigma\rvert^2 N^2)$	$O(\lvert\Sigma\rvert^2 N^3)$	1	1
$(\min,+)$	$O(\lvert\Sigma\rvert^{1.5} N^2)$	$O(\lvert\Sigma\rvert^{1.5} N^3)$	2	1

[1]Note that is is straightforward to modify our protocols to support $(+,\max)$, $(\max,\min)$, and $(\max,+)$.

3 RELATED WORK

In the area of private computational outsourcing there is a growing list of problems considered in this framework (e.g., (Beguin and Quisquater, 1995; Kawamura and Shimbo, 1993; Lim and Lee, 1995; Matsumoto et al., 1988; Pfitzmann and Waidner, 1992; Atallah et al., 2001; Hohenberger and Lysyanskaya, 2005b), and others). We briefly review this work.

In the server-aided secret computation literature (e.g. (Beguin and Quisquater, 1995; Kawamura and Shimbo, 1993; Lim and Lee, 1995; Matsumoto et al., 1988; Pfitzmann and Waidner, 1992; Hohenberger and Lysyanskaya, 2005b), to list a few), a weak smartcard performs public key decryptions by "borrowing" computing power from an untrusted server, without revealing to that server its private information. These papers deal primarily with the important problem of modular exponentiations.

In the privacy homomorphism approach proposed in (Rivest et al., 1978), the outsourcing agent is used as a permanent repository of data, performing certain operations on it and maintaining certain predicates while it remains encrypted, whereas the customer needs only to decrypt the data from the agent to obtain the real data. Our secure outsourcing framework differs in that the customer is not interested in keeping data permanently with the external agents, instead, the customer only wants to temporarily use their superior computational power.

The paper (Abadi et al., 1987) shows an impossibility result: That one cannot hope to outsource an exponential computation while locally doing only polynomial time work. This delineates the limits of what can be achieved. But securely outsourcing expensive polynomial time computations (such as a cubic-time matrix multiplication) remains an important practical goal. Mobile devices can be so computationally weak or battery-limited as to be unable of carrying out computations that would be completely tractable for a powerful server or supercomputer, and therefore would benefit from the development of techniques for the secure outsourcing of expensive polynomial-time computations.

The work in (Atallah et al., 2001) has a different framework because it allows leakage of private information, whereas in this paper we are interested in achieving no leakage.

The paper (Atallah and Li, 2005) used homomorphic encryption for the problem of computing the edit distance between two sequences. Its protocol requires communication between the two servers, a drawback that the framework of the present paper avoids.

There are several protocols in the area of private outsourcing of traditional matrix multiplication (Benjamin and Atallah, 2008; Atallah and Frikken, 2010; Mohassel, 2011). In (Benjamin and Atallah, 2008) the scheme inherently requires more than one server, requires the servers to do expensive homomorphic encryptions, and is vulnerable to collusion by the servers (they would learn the inputs). The scheme in (Atallah and Frikken, 2010) only uses a single server, and uses a heuristic security approach. That is the client has to perform $O(k^2n^2)$ work for a security parameter k. Finally, the scheme in (Mohassel, 2011) considers security in a more traditional approach for single server solutions, but utilizes expensive cryptography.

There are also general results that solve this problem using a single-server. For example, if privacy is the only concern, then fully homomorphic encryption (for example (Gentry, 2009)) could be used to solve this problem.

While there are many protocols for private outsourcing of computations, to our knowledge this paper is the first that considers the general forms of matrix multiplication. Furthermore, as described in the previous section the approaches described here can be used in many settings, including: work in the case of somewhat fully homomorphic encryption, the heuristic approaches of(Atallah and Frikken, 2010), and the multi-party outsourcing based on SMC solutions (such as Sharemind).

4 PROTOCOLS FOR $(+, MIN)$ MATRIX MULTIPLICATION

4.1 A Preliminary Solution

In this section, we propose a protocol for $(+, \min)$ where the workload is $O(|\Sigma|N^2)$ for client and $O(|\Sigma|N^3)$ for server. To simplify the presentation of this protocol, we assume the values are in the range $[1, |\Sigma|]$ (as opposed to $[0, |\Sigma| - 1]$). The crux of this solution are the following two expansions from values in Σ to vectors with $2|\Sigma|$ elements:

1. $T1(x) = x_1, \ldots, x_{|\Sigma|}, x_{|\Sigma|+1}, \ldots, , x_{2|\Sigma|}$ where for $i \in [1, |\Sigma|]$, $x_i = 1$ if $i \leq x$ and is 0 otherwise. Also, for $i \in [|\Sigma| + 1, 2|\Sigma|]$ $x_i = x$ if $i - |\Sigma| = x$ and is 0 otherwise.
2. $T2(x) = x_1, \ldots, x_{|\Sigma|}, x_{|\Sigma|+1}, \ldots, , x_{2|\Sigma|}$ where for $i \in [|\Sigma| + 1, 2|\Sigma|]$, $x_i = 1$ if $i - |\Sigma| < x$ and is 0 otherwise. Also, for $i \in [1, |\Sigma|]$ $x_i = x$ if $i = x$ and is 0 otherwise.

Lemma 1. *Given any two values $x, y \in [1, \Sigma]$, $T1(x) \cdot T2(y) = \min\{x, y\}$.*

Proof: Now, $T1(x) \cdot T2(y) = \left(\sum_{i=1}^{|\Sigma|} x_i y_i\right) + \left(\sum_{i=|\Sigma|+1}^{2|\Sigma|} x_i y_i\right)$. Furthermore, since $y_i = 0$ except where $i = y$, then $\left(\sum_{i=1}^{|\Sigma|} x_i y_i\right) = x_y * y_y = x_y * y$. Now $x_y = 1$ if $x \geq y$ and thus this value is y if $x \geq y$ and is 0 otherwise. A symmetrical argument can be used to show that $\left(\sum_{i=|\Sigma|+1}^{2|\Sigma|} x_i y_i\right) = x$ if $x < y$ and is 0 otherwise. Therefore, $T1(x) \cdot T2(y) = \min\{x, y\}$. □

The protocol is as follows:

1. The client expands A into an $N \times 2|\Sigma|N$ matrix by expanding each cell in its row using $T1$. Also, the client expands B into an $2|\Sigma|N \times N$ matrix by expanding each cell in its column using $T2$.
2. The client outsources the computations $C = AB$, which is the desired result.

For each entry in the matrix, the client performs $O(|\Sigma|)$ work. Thus the client performs work proportional to $O(|\Sigma|N^2)$ total work. The server must perform $O(|\Sigma|N^3)$ total work. Furthermore, the number of rounds is 1, since everything can be done in parallel. Finally, the multiplicative depth is 1.

4.2 Improved Solution

In this section, we propose a protocol for $(+, \min)$. The workload is $O(\sqrt{|\Sigma|}N^2)$ for client and $O(\sqrt{|\Sigma|}N^3)$ for server. The main idea of this approach is that the scheme partitions the values in Σ into groups of size $\sqrt{|\Sigma|}$, and it handles things in the same group differently then it handles things in other groups.

More specifically, Let $\sigma = \sqrt{|\Sigma|}$, then we partition the range $[0, |\Sigma|)$ into σ partitions each of size σ. For any value $v \in [0, |\Sigma|)$, it could be represented by two values less than σ, $v'' = v \bmod \sigma$ and $v' = \lfloor \frac{v}{\sigma} \rfloor$. Note that $v = v'\sigma + v''$. Given a and b, we say that these values are in the same group if $a' = b'$.

First we define 6 expansion functions that expand $v \in [0, |\Sigma|)$ into vectors of length σ.

1. $V1(v) = f_0, \ldots, f_{\sigma-1}$ where $f_i = 1$ if $i < v'$ and is 0 otherwise.
2. $V2(v) = f_0, \ldots, f_{\sigma-1}$ where $f_i = v$ if $i = v'$ and is 0 otherwise.
3. $V3(v) = f_0, \ldots, f_{\sigma-1}$ where $f_i = 1$ if $i = v'$ and is 0 otherwise.
4. $V4(v) = f_0, \ldots, f_{\sigma-1}$ where $f_i = 1$ if $i \leq v''$ and is 0 otherwise.
5. $V5(v) = f_0, \ldots, f_{\sigma-1}$ where $f_i = 1$ if $i < v''$ and is 0 otherwise.
6. $V6(v) = f_0, \ldots, f_{\sigma-1}$ where $f_i = v$ if $i = v''$ and is 0 otherwise.

Lemma 2. *For any values $a, b \in \Sigma$, $(V3(a) \cdot V3(b))\ (V4(a) \cdot V6(b) + V5(b) \cdot V6(a)) + (V1(a) \cdot V2(b) + V1(b) \cdot V2(a)) = \min\{a, b\}$ where $\cdot$ denotes the inner product of vectors*

Proof:
There are two cases to consider: i) the case where a and b are in different groups, and ii) the case where a and b are in the same group.

Consider the case where a and b are in different groups. Now, $V1(a) \cdot V2(b) = b$ if $b' < a'$ and is 0 otherwise. Furthermore, $V1(b) \cdot V2(a) = a$ if $a' < b'$ and is 0 otherwise. Hence, If a and b are in different groups, then $V1(a) \cdot V2(b) + V1(b) \cdot V2(a) = \min\{a, b\}$. If a and b are in the same group, then this is 0.

Consider the case where a and b are in the same group. First, $V3(a) \cdot V3(b) = 1$ if $a' = b'$ and is 0 otherwise. That is, this value is 1 if a and b are in the same group and is 0 otherwise. Given that $a' = b'$ then:

1. If $a \geq b$, then $V4(a) \cdot V6(b) = b$. Otherwise, $V4(a) \cdot V6(b) = 0$
2. If $a < b$, then $V5(b) \cdot V6(a) = a$. Otherwise, $V5(b) \cdot V6(a) = 0$

Putting these two pieces together: if $a' = b'$ then $(V4(a) \cdot V6(b) + V5(b) \cdot V6(a)) = \min\{a, b\}$. Hence, $(V3(a) \cdot V3(b))(V4(a) \cdot V6(b) + V5(b) \cdot V6(a)) = \min\{a, b\}$ when a and b are in the same group, but is 0 if there are in different groups.

Therefore by combining these two cases: $(V3(a) \cdot V3(b))(V4(a) \cdot V6(b) + V5(b) \cdot V6(a)) + (V1(a) \cdot V2(b) + V1(b) \cdot V2(a)) = \min\{a, b\}$. □

The protocol is as follows:

1. The client creates 5 matrices A_i (resp. B_j), each cell of which is the expansion vector Vi (respectively, Vj) of the corresponding cell in A (resp. in B), for $i = 1, 2, 3, 4, 6$ (resp. $j = 1, 2, 3, 5, 6$.)
2. The client uses an outsourced server to compute a matrix C where $C[i, j] = \sum_{\ell=1}^{N}((A_3(i, \ell) \cdot B_3(\ell, j))(A_4(i, \ell) \cdot B_6(\ell, j) + B_5(\ell, j) \cdot A_6(i, \ell)) + (A_1(i, \ell) \cdot B_2(\ell, j) + B_1(\ell, j) \cdot A_2(i, \ell)))$.

C is the $(+, \min)$ product of A and B, because each cell of C is the sum of N minimums of aligned cells in A and B. The client's work is mainly to create matrices A_i and B_j, which is of size $O(\sigma N^2) = O(\sqrt{|\Sigma|}N^2)$. And the server's work is $O(\sqrt{|\Sigma|}N^3)$ to compute the N^2 result cells, since for each cell he has to compute the sums of N minimums by the circuit. Notice that the multiplicative depth of this circuit is 2, and number of communication rounds is 1.

5 PROTOCOLS FOR $(\min, \max)$ MATRIX MULTIPLICATION

5.1 Preliminary Solution

We now introduce a protocol for $(\min, \max)$. To achieve this the client will outsource $|\Sigma|$ matrix multiplications in parallel (each of which has multiplicative depth of 1). The ith multiplication will reveal which entries in the result matrix are $\leq i$. These values can be combined together to find the actual values of the entries; that is, the value of a cell is the minimum matrix product that is non-zero in that cell.

The protocol is as follows:

1. For each $i \in [0, |\Sigma)$, the client computes A_i and B_i which are $N \times N$ matrices where $A_i[j,k]$ (resp. $B_i[j,k]$) is 1 if $A[j,k] \leq i$ (resp. $B[j,k] \leq i$) and is 0 otherwise. Note that $A_i[j,k] * B_i[k,\ell] = 1$ if both $A_i[j,k]$ and $B_[k,\ell$ are $\leq i$. In other words, $A_i[j,k] * B_i[k,\ell] = 1$ if $\max\{A[j,k], B[k,\ell]\} \leq i$ and is 0 otherwise.
2. The client has the server compute $C_i = A_i B_i$ for all $i \in [0, |\Sigma|)$. Note that $C_i[j,k] = \sum_{\ell=1}^{N} (A_i[j,\ell] * B_i[\ell,k])$, which will be non-zero if and only if there is an ℓ such that $A[j,k]$ and $B[k,\ell]$ are both $\leq i$. Such an ℓ exists if and only if $\min_{z \in [1,N]}\{\max\{A[j,z], B[z,k]\}\} \leq i$.
3. The client computes the result matrix C, by setting $C[i,j]$ to be the smallest value ℓ such that $C_\ell[i,j]$ is non-zero.

The above protocol works because $C_i[j,k]$ will be non-zero if and only if there is a value ℓ such that $A_i[j,\ell]$ and $B_i[\ell,k]$ are both 1. But this means that $A_i[j,\ell]$ and $B_i[\ell,k]$ are both $\leq i$, and thus the minimum of the maximums will be $\leq i$.

Note that the client has to perform $O(|\Sigma|N^2)$ work, and the server has to perform $O(|\Sigma|N^3)$ work. The number of rounds is 1, and the multiplicative depth is 1.

5.2 Improved Solution

In this section, we improve the protocol in the previous section by reducing the client's cost to $O(\sqrt{|\Sigma|}N^2)$. This is achieved by increasing the depth of the circuit and by increasing the number of rounds. Let $\sigma = \sqrt{|\Sigma|}$, and given a value $v \in [0, |\Sigma|)$, let $v'' = v \bmod \sigma$ and $v' = \lfloor \frac{v}{\sigma} \rfloor$. Note that $v = v'\sigma + v''$. We refer to v' as the group of v and to v'' as the offset of v.

First observe that the protocol in the previous section can be modified to find the value of each cell within an additive factor of σ. That is, it is possible to find a matrix C' where $C'[i,j] = c$ if $C[i,j] \in [c\sigma, (c+1)\sigma)$. Furthermore, this can be done by compute A_i and B_i only for multiples of σ in the protocol in the previous section. The client's cost of this protocol is $O(\sigma N^2)$. This will be the first round of the computation, and this matrix will be used in the second round.

The second part of this protocol will be to compute the offset in each group. The difficult part will be creating a test for each cell of the matrix that tests if the offset is $\leq t$ for some value t. The difficult part of this is to ignore the pairs of values that are not in the correct group for a cell.

Before giving our full protocol, we describe a few functions that are used in the construction.

1. Given $v \in [0, \sigma)$, $EQ(v) = v_0, \ldots, v_\sigma$ where $v_v = 1$ and all other values are 0.
2. Given $v \in [0, \sigma)$, $LT(v) = v_0, \ldots, v_\sigma$ where $v_i = 1$ if $i < v$ and is 0 otherwise.
3. Given $v \in [0, \sigma)$, $EQ_t(v) = 1$ if $t = v$ and is 0 otherwise.
4. Given $v \in [0, \sigma)$, $LT_t(v) = 1$ if $v < t$ and is 0 otherwise.
5. Given $v \in [0, \sigma)$, $LE_t(v) = 1$ if $v \leq t$ and is 0 otherwise.

Given values $a, b \in [0, \Sigma)$ and $c, t \in [0, \sigma)$ suppose that we wanted to create a test to determine if $\max\{a,b\} = c\sigma + t$. There are 3 possible ways that this could happen: 1) $a' < c$ and $b = c\sigma + t$, 2) $b' < c$ and $a = c\sigma + t$, and 3) $a' = c$, $b' = c$, and $\max\{a'', b''\} = t$. Note that these three cases are mutually exclusive. We now define various functions that are used in the protocol:

1. $F1_t(a,b,c) = (EQ(a') \cdot LT(c)) * (EQ(b') \cdot EQ(c)) * EQ_t(b'')$. Note that $F1_t(a,b,c) = 1$ if $a' < c$ and $b = c\sigma + t$ and is 0 otherwise. Also, note that this function has multiplicative depth 3.
2. $F2_t(a,b,c) = (EQ(b') \cdot LT(c)) * (EQ(a') \cdot EQ(c)) * EQ_t(a'')$. Note that $F2_t(a,b,c) = 1$ if $b' < c$ and $a = c\sigma + t$ and is 0 otherwise. Also, note that this function has multiplicative depth 3.
3. $F_3(t,a,b,c) = (EQ(a') \cdot EQ(c)) * (EQ(b') \cdot EQ(c)) * (LE_t(b'')EQ_t(a'') + LT_t(a'')EQ_t(b''))$. Note that $F3_t(a,b,c) = 1$ if $a' = b' = c$ and $\max\{a,b\} = c\sigma + t$ and is 0 otherwise. Also, note that this function has multiplicative depth 3.
4. $F_t(a,b,c) = F1_t(a,b,c) + F2_t(a,b,c) + F3_t(a,b.c)$. Note that this function is 1 if $\max\{a,b\} = c\sigma + t$ and is 0 otherwise. The three cases above are mutually exclusive and partition

all possible cases where $\max\{a,b\} = c\sigma + t$. Note that this function has multiplicative depth 3. It is important to note that to compute this function that we need only to have: $LT(c), EQ(c), EQ(a'), EQ(b'), EQ_t(b''), EQ_t(a''), LE_t(b''), LT_t(a'')$.

We are now ready to give the protocol.

1. For each $i \in [0,\sigma)$, the client computes $A_{i\sigma}$ and $B_{i\sigma}$ which are $N \times N$ matrices where $A_i[j,k]$ (resp. $B_i[j,k]$) is 1 if $A[j,k] \leq i$ (resp. $B[j,k] \leq i$) and is 0 otherwise.
2. The client has the server compute $C_{i\sigma} = A_{i\sigma}B_{i\sigma}$ for all $i \in [0,\sigma)$. These can be done in parallel.
3. The client computes an intermediate matrix C', by setting $C'[i,j]$ to be the smallest value ℓ such that $C_{\ell\sigma}[i,j]$ is non-zero.
4. For each $t \in [0,\sigma)$, the client outsources the computations of a matrix C_t where $C_t[i,j] = \sum_{\ell=1}^{N} F_t(A[i,\ell], B[\ell,j], C'[i,j])$. This is done by having the client outsource the above mentioned vectors and values to the server(or servers) that are necessary for this computation. Note that $C_t[i,j]$ is the number of values $\ell \in [1,N]$ such that $\max\{A[i,\ell], B[\ell,j]\} = C'[i,j]\sigma + t$.
5. Given $C_0,\ldots,C_{\sigma-1}$, the client computes the result $C[i,j]$ as follows: Let q be the smallest values such that $C_q[i,j]$ is non-zero and set $C[i,j] = C'[i,j]\sigma + q$.

The above protocol requires the client to perform $O(\sqrt{|\Sigma|}N^2)$ computations, the server to perform $O(\sqrt{|\Sigma|}N^3)$ computations, requires multiplicative depth 3 and requires 2 rounds.

6 PROTOCOLS FOR $(MIN,+)$ MATRIX MULTIPLICATION

6.1 Preliminary Solution

In this section we present a scheme for $(\min,+)$ where the client performs work $O(|\Sigma|^2N^2)$. The main idea of this scheme is to perform a test for each possible value of a cell in the result. The test for value T will determine which cells in the result have a value $\leq T$ (specifically the test counts the number of values in the corresponding row of A and column of B where the sum is $\leq T$). This test is performed for all values from 0 to $2|\Sigma|-2$. These results can be combined together to determine the actual value of the cell by finding the minimum value of T where the value is non-zero for each cell. We now give the functionality $TEST_T(A,B)$ (which is used to test if a cell in the result is $\leq T$):

1. For each cell $A[i,j]$, expand it into $|\Sigma|$ cells in its row. Denote these cells as $A^{(0)}[i,j],\ldots,A^{(|\Sigma|-1)}[i,j]$ where $A^{(\ell)}[i,j] = 1$ if $\ell + A[i,j] \leq T$ and is 0 otherwise. Denote as A_T the $N \times |\Sigma|N$ matrix that results from this expansion.
2. For each cell $B[i,j]$, expand it into $|\Sigma|$ cells in its column. Denote these cells as $B^{(0)}[i,j],\ldots,B^{(|\Sigma|-1)}[i,j]$ where $B^{(\ell)}[i,j] = 1$ if $\ell = B[i,j]$ and is 0 otherwise. Denote as B_T the $|\Sigma|N \times N$ matrix that results from this expansion.

As an example, suppose that $|\Sigma| = 4$, $A[i,j] = 2$ and $B[j,k] = 1$. When doing the test for $T = 3$, the value of $A[i,j]$ will be expanded into $A^{(0)}[i,j],\ldots,A^{(3)}[i,j] = 1,1,0,0$, and the value of $B[j,k]$ will be expanded into $B^{(0)}[i,j],\ldots,B^{(0}[i,j] = 0,1,0,0$. Notice that the scalar product of these two values is 1, and in this case $A[i,j] + B[j,k] \leq 3$. More generally, the scalar product of the expansions of two values will be 1 when the sum of the two values is $\leq T$ and and is 0 otherwise.

If we denote as $C_T = A_T B_T$ where $(A_T, B_T) \leftarrow TEST_T(A,B)$, then C_T is a $N \times N$ matrix with the following property:

Lemma 3. *$C_T[i,j] = 0$ if and only if $\min\{A[i][\ell] + B[\ell][j] : 1 \leq \ell \leq N\} > T$.*

Proof: Now, since $C_T = A_T B_T$:

$$C_T[i,j] = \sum_{\ell=1}^{N} \sum_{m=0}^{|\Sigma|-1} A_T^{(m)}[i,\ell] B_T^{(m)}[\ell,j]$$

According to the transformation $B_T^{(m)}[\ell,j] = 1$ if $m = B[\ell,j]$ and is 0 otherwise, hence $A_T^{(m)}[i,\ell]B_T^{(m)}[\ell,j] = 0$ if $m \neq B[\ell,j]$, and is $A_T^{(m)}[i,\ell]$ otherwise. Therefore,

$$\sum_{\ell=1}^{N} \sum_{m=0}^{|\Sigma|-1} A_T^{(m)}[i,\ell] B_T^{(m)}[\ell,j] = \sum_{\ell=1}^{N} A_T^{(B[\ell,j])}[i,\ell]$$

According to the transformation $A_T^{(B[\ell,j])}[i,\ell] = 1$ if $A[i,\ell] + B[\ell,j] \leq T$ and is 0 otherwise. Hence, $C_T[i,j] = 0$ if and only if $A[i,\ell] + B[\ell,j] > T$ for all $\ell \in [0,N-1]$. However, this means that $\min\{A[i][\ell] + B[\ell][j] : 0 \leq \ell \leq N-1\} > T$, which concludes the proof. □

Now that the $TEST_T$ functionality has been defined, we present the main protocol of this section for computing the $(\min,+)$ matrix multiplication of A and B:

1. The data owner computes $(A_i, B_i) \leftarrow TEST_i(A, B)$ for all $i \in [0, 2|\Sigma| - 2]$.
2. The data owner uses an outsourced server to compute $C_i = A_i B_i$ for all $i \in [0, 2|\Sigma| - 2]$.
3. Set $C[i, j]$ to be the smallest value v such that $C_v[i, j] \neq 0$.

The correctness of the above algorithm is based on Lemma 3. That is, if $C_{v-1}[i, j] = 0$ and $C_v[i, j] \neq 0$, then there must be at least one value in $A[i, \ell] + B[\ell, j] \leq v$, but there is no such value where $A[i, \ell] + B[\ell, j] \leq v - 1$, and hence the minimum of the sums is v.

The client's work in the $TEST$ functionality is $O(|\Sigma|N^2)$, and the test functionality is invoked $O(|\Sigma|)$ times, and thus the client's work is $O(|\Sigma|^2 N^2)$. Furthermore for each $TEST$ the server must perform $O(|\Sigma|N^3)$ work, and thus the server's computation is $O(|\Sigma|^2 N^3)$. The number of rounds in the protocol is 1, since the matrix multiplications can be done in parallel. Finally, the server only performs matrix multiplications, and thus multiplication depth is 1.

6.2 Reducing Costs by Increasing Depth

In this section we give a scheme that has client complexity $O(|\Sigma|^{1.5} N^2)$ for $(\min, +)$. The main idea of this scheme is that it trades off lower computational costs by increasing the multiplicative depth of the circuits that are used. Essentially, this scheme reduces the cost of the test functionality in the previous algorithm to $O(\sqrt{\Sigma}|)$.

Let $\sigma = \sqrt{|\Sigma|}$, and given a value $v \in [0, |\Sigma|)$, let $v'' = v \bmod \sigma$ and $v' = \lfloor \frac{v}{\sigma} \rfloor$. Note that $v = v'\sigma + v''$.

Before describing our scheme we define several expansion functions that takes a value $v \in [0, |\Sigma|)$ and a value $t \in [0, 2|\Sigma| - 2]$ and produces a vector with σ elements.

1. $V1(t, v) = a_0, \ldots, a_{\sigma-1}$ where $a_i = 1$ if $i = t' - v' - 1$ and is 0 otherwise.
2. $V2(t, v) = a_0, \ldots, a_{\sigma-1}$ where $a_i = 1$ if $i = t' - v'$ and is 0 otherwise.
3. $V3(t, v) = a_0, \ldots, a_{\sigma-1}$ where $a_i = 1$ if $i = t'' - v''$ and is 0 otherwise.
4. $V4(t, v) = a_0, \ldots, a_{\sigma-1}$ where $a_i = 1$ if $i = \sigma + t'' - v''$ and is 0 otherwise.
5. $V5(t, v) = a_0, \ldots, a_{\sigma-1}$ where $a_i = 1$ if $i = v'$ and is 0 otherwise.
6. $V6(t, v) = a_0, \ldots, a_{\sigma-1}$ where $a_i = 1$ if $i = v''$ and is 0 otherwise.

We will test when a specific cell has any cells in its corresponding row and column where the sum of the cells is equal to a target value t. Given two values a and b, there are two mutually exclusive ways in which this can happen: 1) $a' + b' = t'$ and $a'' + b'' = t''$ and 2) $a' + b' = t' - 1$ and $a'' + b'' = \sigma + t''$. We consider each of these cases separately:

For the first case the value $V2(t, a) \cdot V_5(t, b) = 1$ if $a' + b' = t'$ and is 0 otherwise. To see this, suppose that $V2(t, a) = a_0, \ldots, a_{\sigma-1}$ and $V5(t, b) = b_0, \ldots, b_{\sigma-1}$. Now $b_i = 1$ only when $i = b'$, and thus $V2(t, a) \cdot V_5(t, b) = a_{b'}$, which will be 1 iff $b' = t' - a'$. Consider, the value $V_3(t, a) \cdot V6(t, b)$; this value will be 1 if and only if $a'' + b'' = t''$. To see this, notice that both of these vectors have a single 1, and so the ones must align. Furthermore, these values align only when $a'' + b'' = t''$. Thus if we compute $(V2(t, a) \cdot V_5(t, b)) * (V_3(t, a) \cdot V6(t, b))$, then this value will be 1 only when $a' + b' = t'$ and $a'' + b'' = t''$ and will be 0 otherwise.

Now consider the second case. The value $V1(t, a) \cdot V5(t, b)$ will be 1 only when $a' + b' = t' - 1$ and will be 0 otherwise. The reasoning is the same as in the previous paragraph. Furthermore, the values $V4(t, a) \cdot V6(t, b)$ is 1 only when $a'' + b'' = \sigma + t''$ and will be 0 otherwise. Thus if we compute $(V1(t, a) \cdot V_5(t, b)) * (V_4(t, a) \cdot V6(t, b))$, then this value will be 1 only when $a' + b' = t' - 1$ and $a'' + b'' = \sigma + t''$ and will be 0 otherwise.

Putting these pieces together, the value $(V2(t, a) \cdot V_5(t, b)) * (V_3(t, a) \cdot V6(t, b)) + (V1(t, a) \cdot V_5(t, b)) * (V_4(t, a) \cdot V6(t, b))$ will be 1 if $a + b = t$ and will be 0 otherwise. As a shorthand notation we denote this value as $F(a, b, t)$.

We are now ready to present the main protocol that will use the above vectors and values.

1. For each value $t \in [0, 2|\Sigma| - 2]$ the client does the following:
   (a) For each entry $A[i, j]$, the client computes $V1(t, A[i, j])$, $V2(t, A[i, j])$, $V3(t, A[i, j])$, and $V4(t, A[i, j])$.
   (b) For each entry $B[i, j]$, the client computes $V5(t, B[i, j])$, $V6(t, B[i, j])$
   (c) The client has the server compute a matrix C_t where $C_t[i, j] = \sum_{\ell=1}^{N} F(A[i, \ell], B[\ell, j], t)$.
2. For the end result, the client sets the value of $C[i, j]$ to be the smallest value t such that $C_t[i, j] \neq 0$.

The above protocol works because $F(v_1, v_2, t)$ is non-zero only when $v_1 + v_2 = t$. Thus $\sum_{\ell=1}^{N} F(A[i, \ell], B[\ell, j], t)$ will be non-zero if and only if there is a pair of entries in the ith row in A and jth column of B such that the sum of the two values is t. Hence, the minimum value t such that $C_t[i, j]$ is non-zero will be the minimum of the sums as required.

For each entry in the matrix, the client performs $O(\sigma)$ work per test. Since there are $O(|\Sigma|)$ such tests the client performs work proportional to $O(|\Sigma|^{1.5}N^2)$ total work. The server must perform $O(|\Sigma|^{1.5}N^3)$ total work. Furthermore, the number of rounds is 1, since everything can be done in parallel. Finally, the multiplicative depth is 2.

7 CONCLUSIONS AND FUTURE WORK

We introduced techniques for outsourcing matrix multiplications over closed semi-rings and other algebraic structures that lack the ring algebraic structure of the $(+,*)$ multiplication. These techniques can be used in a variety of settings to outsource the computations in a secure manner. The remote server being used does not learn anything about the inputs or the outputs to a computation other than the size of the matrices and of the alphabet. Although in most practical situations it is harmless to leak the size of the alphabet, it is nevertheless of intellectual interest to extend our protocols so as to hide it. There is a more practical need for extending our schemes to hide the size of the matrices, which we leave for future work.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their comments and useful suggestions. Portions of this work were supported National Science Foundation Grants CNS-0915436, CNS-0913875,CNS-0915843, Science and Technology Center CCF-0939370; by an NPRP grant from the Qatar National Research Fund; by Grant FA9550-09-1-0223 from the Air Force Office of Scientific Research; and by sponsors of the Center for Education and Research in Information Assurance and Security. The statements made herein are solely the responsibility of the authors.

REFERENCES

Abadi, M., Feigenbaum, J., and Kilian, J. (1987). On hiding information from an oracle. In *Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 195–203. ACM Press.

Atallah, M. J. and Frikken, K. B. (2010). Securely outsourcing linear algebra computations. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, pages 48–59, New York, NY, USA. ACM.

Atallah, M. J. and Li, J. (2005). Secure outsourcing of sequence comparisons. In *International Journal of Information Security*, pages 277–287.

Atallah, M. J., Pantazopoulos, K. N., Rice, J., and Spafford, E. H. (2001). Secure outsourcing of scientific computations. *Advances in Computers*, 54(6):215–272.

Beguin, P. and Quisquater, J. J. (1995). Fast server-aided rsa signatures secure against active attacks. In *CRYPT0 95*, pages 57–69.

Benjamin, D. and Atallah, M. J. (2008). Private and cheating-free outsourcing of algebraic computations. In *Sixth Annual Conference on Privacy, Security and Trust, PST 2008, October 1-3, 2008, Fredericton, New Brunswick, Canada*, pages 240–245.

Bogdanov, D., Laur, S., and Willemson, J. (2008). Sharemind: A framework for fast privacy-preserving computations. In Jajodia, S. and Lopez, J., editors, *Computer Security - ESORICS 2008*, volume 5283 of *Lecture Notes in Computer Science*, pages 192–206. Springer Berlin / Heidelberg.

Boneh, D., Goh, E.-J., and Nissim, K. (2005). Evaluating 2-dnf formulas on ciphertexts. In Kilian, J., editor, *Theory of Cryptography*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer Berlin / Heidelberg.

Brakerski, Z. and Vaikuntanathan, V. (2011a). Efficient fully homomorphic encryption from (standard) lwe. Cryptology ePrint Archive, Report 2011/344.

Brakerski, Z. and Vaikuntanathan, V. (2011b). Fully homomorphic encryption from ring-lwe and security for key dependent messages. In Rogaway, P., editor, *Advances in Cryptology CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer Berlin / Heidelberg.

Cormen, T. H., Stein, C., Rivest, R. L., and Leiserson, C. E. (2001). *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition.

Duan, R. and Pettie, S. (2009). Fast algorithms for (max, min)-matrix multiplication and bottleneck shortest paths. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, pages 384–391, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Felzenszwalb, P. and McAuley, J. (2011). Fast inference with min-sum matrix product. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(12):2549 –2554.

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, STOC '09, pages 169–178, New York, NY, USA. ACM.

Goldreich, O., Micali, S., and Wigderson, A. (1987). How to play any mental game. In *Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 218–229.

Hohenberger, S. and Lysyanskaya, A. (2005a). How to securely outsource cryptographic computations. In Kilian, J., editor, *Theory of Cryptography*, volume 3378

of *Lecture Notes in Computer Science*, pages 264–282. Springer Berlin / Heidelberg.

Hohenberger, S. and Lysyanskaya, A. (2005b). How to securely outsource cryptographic computations. In *Theory of Cryptography Conference (TCC'05)*, volume 3378 of *LNCS*, pages 264–282.

Kawamura, S. I. and Shimbo, A. (1993). Fast server-aided secret computation protocols for modular exponentiation. *IEEE Journal on Selected Areas in Communications*, 11(5):778–784.

Lim, C. H. and Lee, P. J. (1995). Security and performance of server-aided rsa computation protocols. In *CRYPT0 95*, pages 70–83.

Matsumoto, T., Kato, K., and Imai, H. (1988). Speeding up secret computations with insecure auxiliary devices. In *CRYPT0 88*, pages 497–506.

Mohassel, P. (2011). Efficient and secure delegation of linear algebra. Cryptology ePrint Archive, Report 2011/605. http://eprint.iacr.org/.

Naehrig, M., Lauter, K., and Vaikuntanathan, V. (2011). Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM workshop on Cloud computing security workshop*, CCSW '11, pages 113–124, New York, NY, USA. ACM.

Pfitzmann, B. and Waidner, M. (1992). Attacks on protocols for server-aided rsa computations. In *EUROCRYPT 92*, pages 153–162.

Rivest, R. L., Adleman, L., and Dertouzos, M. L. (1978). On data banks and privacy homomorphisms. *Foundations of Secure Computation*, pages 169–177.

Saks, M. E. (1988). A limit theorem for (min, +) matrix multiplication. *Math. Oper. Res.*, 13:606–618.

Vassilevska, V., Williams, R., and Yuster, R. (2007). All-pairs bottleneck paths for general graphs in truly subcubic time. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, STOC '07, pages 585–589, New York, NY, USA. ACM.