Achieving Fairness in Private Contract Negotiation [⋆]

Keith Frikken and Mikhail Atallah
CERIAS and Department of Computer Sciences

Purdue University

Abstract. Suppose Alice and Bob are two entities (e.g. agents, organizations, etc.) that wish to negotiate a contract. A contract consists of several clauses, and each party has certain constraints on the acceptability and desirability (i.e., a private "utility" function) of each clause. If Bob were to reveal his constraints to Alice in order to find an agreement, then she would learn an unacceptable amount of information about his business operations or strategy. To alleviate this problem we propose the use of Secure Function Evaluation (SFE) to find an agreement between the two parties. There are two parts to this: i) determining whether an agreement is possible (if not then no other information should be revealed), and ii) in case an agreement is possible, coming up with a contract that is *valid* (acceptable to both parties), *fair* (when many valid and good outcomes are possible one of them is selected randomly with a uniform distribution, without either party being able to control the outcome), and *efficient* (no clause is replaceable by another that is better for both parties). It is the fairness constraint in (ii) that is the centerpiece of this paper as it requires novel techniques that produce a solution that is more efficient than general SFE techniques. We give protocols for all of the above in the semi-honest model, and we do not assume the Random Oracle Model.

1 Introduction

Suppose Alice and Bob are two entities who are negotiating a joint contract, which consists of a sequence of clauses (i.e., terms and conditions). Alice and Bob are negotiating the specific value for each clause. Example clauses include:

1. How will Alice and Bob distribute the revenue received for jointly performing a task?
2. Given a set of tasks, where Alice and Bob each have a set of tasks they are willing and able to perform, who performs which tasks?

[⋆] Portions of this work were supported by Grants EIA-9903545, IIS-0219560, IIS-0312357, and IIS-0242421 from the National Science Foundation, Contract N00014-02-1-0364 from the Office of Naval Research, by sponsors of the Center for Education and Research in Information Assurance and Security, and by Purdue Discovery Park's e-enterprise Center.

3. Given a set of locations to perform certain tasks, in which locations does Alice (respectively, Bob) perform their tasks?

Alice and Bob will each have private constraints on the acceptability of each clause (i.e., rules for when a specific term is acceptable). A specific clause is an agreement between Alice and Bob if it satisfies both of their constraints. In a non-private setting, Alice and Bob can simply reveal their constraints to one another. However, this has two significant drawbacks: i) if there are multiple possible agreements how do Alice and Bob choose a specific agreement (some are more desirable to Alice, others more desirable to Bob), and ii) the revelation of one's constraints and preferences is unacceptable in many cases (e.g., if one's counterpart in the negotiation can use these to infer information about one's strategies or business processes or even use them to gain an information advantage for use in a future negotiation). This second problem is exacerbated when Alice and Bob are competitors in one business sector but cooperate in another sector. We propose a framework and protocols that facilitate contract negotiation without revealing private constraints on the contract. There are two components to such a negotiation: i) the ability to determine if there is a contract that satisfies both parties' constraints (without revealing anything other than "yes/no") and ii) if there is a contract that satisfies both parties' constraints, then a protocol for determining a contract that is *valid* (acceptable to both parties), *fair* (when many valid and good outcomes are possible one of them is selected randomly with a uniform distribution, without either party being able to control the outcome), and *efficient* (no clause is replaceable by another that is better for both parties).

We introduce protocols for both of these tasks in the semi-honest model (i.e., the parties will follow the protocol steps but may try to learn additional information). The results of the paper are summarized as follows:

– The definition of a framework for privacy preserving contract negotiation. This framework allows multiple independent clauses, but can be extended to support dependencies between different clauses.
– Protocols for determining if there is a valid contract according to both parties' constraints.
– Protocols for determining a fair, valid, and efficient contract when there is such a contract in the semi-honest model. The most difficult of these requirements is fairness, and we believe that the ability to choose one of several values without either party having control of the value will have applications in other domains.

The rest of the paper is organized as follows. In Section 2, an overview of related work is given. In Section 3, several building blocks are given that are used in later protocols. Section 4 describes our security model. Section 5, outlines the framework for secure contract negotiation. Section 6, describes protocols for computing the satisfiability of a clause as well as for determining a valid term for a clause. Section 7 we discuss extensions to our protocols that allow Alice and Bob to make preferences. Section 8 introduces several extensions to our framework. Finally, Section 9 summarizes the results.

2 Related Work

The authors are not aware of any directly related work in this area. Much of the previous work in automated contract negotiation ([15, 14, 26]) focuses on creating logics to express contract constraints so that agreements can be computed. Our work is not to be confused with simultaneous contract signing [24], which solves the different problem of achieving simultaneity in the signing of a pre-existing already agreed-upon contract. The closest work is in [25], which deals with user preference searching in e-commerce. The problem addressed there is that a vendor may take advantage of a customer if that vendor learns the customer's constraints on a purchase (type of item, spending range, etc.). To prevent this, [25] suggests using a gradual information release.

Secure Multi-party Computation (SMC) was introduced in [27], which contained a scheme for secure comparison; suppose Alice (with input a) and Bob (with input b) desire to determine whether or not $a < b$ and without revealing any information other than this result (this is referred to as "Yao's Millionaire Problem"). More generally, SMC allows Alice and Bob with respective private inputs a and b to compute a function $f(a, b)$ by engaging in a secure protocol for some public function f. Furthermore, the protocol is private in that it reveals no additional information. By this what is meant is Alice (Bob) learns nothing other than what can be deduced from a (b) and $f(a, b)$. Elegant general schemes are given in [11, 10, 1, 4] for computing any function f privately. One of the general results in Two-party SMC is that if given a circuit of binary gates for computing a function f that has m input wires and n gates, then there is a mechanism for securely evaluating the circuit with m chosen 1-out-of-2 Oblivious Transfers(OTs), communication proportional to n, and a constant number of rounds [28]. There have been many extensions of this including: multiple parties, malicious adversaries, adaptive adversaries, and universal protocols [9, 3, 17]. Furthermore, [19] implemented the basic protocol along with a compiler for building secure protocols. However, these general solutions are considered impractical for many problems, and it was suggested in [13] that more efficient domain-specific solutions can be developed.

A specific SMC-based application that is similar to our work is that of [6], which introduced protocols for computing set intersection efficiently. Specifically, it introduced protocols (for the semi-honest and malicious models) for computing the intersection between two sets, the cardinality of set intersection, and to determine if the cardinality was above a threshold. That work also introduced protocols for multiple parties, approximating intersection, and for fuzzy set intersection. This is similar to the centerpiece of this paper as our work can be summarized as "choose a random element of the intersection, given that there is such an element".

3 Building Blocks

3.1 Cryptographic Primitives and Definitions

1. *Oblivious Transfer:* There are many equivalent definitions of Oblivious Transfer (OT), and in this paper we use the definition of chosen 1-out-of-N OT, where Alice has a set of items $x_1, \ldots, x_N$ and Bob has an index $i \in \{1, \ldots, N\}$. The OT protocol allows Bob to obtain x_i without revealing any information about i to Alice and without revealing any information about other x_j $(j \neq i)$ values to Bob. A high-level overview of OT can be found in [24]. Recently, there has been some work on the development of efficient OT protocols [20, 21]. It was also shown in [16] that there was no black-box reduction from OT to one-way functions.
2. *Homomorphic Encryption:* A cryptographic scheme is said to be homomorphic if for its encryption function E the following holds: $E(x) * E(y) = E(x + y)$. Examples of homomorphic schemes are described in [5, 23, 22]. A cryptographic scheme is said to be *semantically secure* if $E(x)$ reveals no information about x. In other words $(E(x), E(x))$ and $(E(x), E(y))$ are computationally indistinguishable (defined in Section 4).
3. *Secure Circuit Evaluation*: A well known result in SMC is that boolean circuits composed of 2-ary gates can be evaluated with communication equal to the size of the circuit, a 1-out-of-2 OT per input wire, and a constant number of rounds [28, 12].

4 Preliminaries

In this section, we discuss our security model, which is similar to that of [2, 9] (however we use a subset of their definitions as we are in the semi-honest model). At a high level a protocol securely implements a function f if the information that can be learned by engaging in the protocol, could be learned in an ideal implementation of the protocol where the functionality was provided by a trusted oracle. We consider semi-honest adversaries (i.e., those that will follow the protocol but will try to compute additional information other than what can be deduced from their input and output alone).

We now formally review the above notions for two party protocols. We do this by defining the notion of an ideal-model adversary (one for the situation where there is a trusted oracle) and a real-model adversary for the protocol Π, and then state that a protocol is secure if the two executions are computationally indistinguishable. Assume that Π computes some function $f : \{0,1\}^\star \times \{0,1\}^\star \rightarrow \{0,1\}^\star$.

Alice (Bob) is viewed as a Probabilistic Polynomial Time (PPT) algorithm A (B) that can be decomposed into two parts A_I and A_O (B_I and B_O). Also Alice's (Bob's) private input is represented by $X_A(X_B)$. We represent Alice's view of the protocol as $\mathit{IDEAL}_{A,B}(X_A, X_B) = (A_O(X_A, r_A, Y_A), f(A_I(X_A, r_A), X_B))$ where r_A is Alice's private coin flips.

We now define the actual execution for a protocol Π that implements the function f. In a real model, the parties are arbitrary PPT algorithms (A', B'). The adversaries are admissible if both parties use the algorithm specified by protocol Π (as we are in the semi-honest model). We denote the interaction of protocol Π by $\mathit{REAL}_{\Pi,A',B'}(X_A, X_B)$, which is the output from the interaction of $A'(X_A)$ and $B'(X_B)$ for protocol Π.

As is usual, we say that a protocol Π securely evaluates a function f if for any admissible adversary in the real model (A', B'), there exists an ideal-model adversary (A, B) such that $\mathit{IDEAL}_{A,B}(X_A, X_B)$ and $\mathit{REAL}_{\Pi,A',B'}(X_A, X_B)$ are computationally indistinguishable. To define what is meant by this we recall the standard definition of computational indistinguishability [8]: Two probability ensembles $X \overset{\text{def}}{=} \{X_n\}_{n\in\mathcal{N}}$ and $Y \overset{\text{def}}{=} \{Y_n\}_{n\in\mathcal{N}}$ are computationally indistinguishable if for any PPT algorithm D, any polynomial p, and sufficiently large n it holds that:

$|(Pr(D(X_n, 1^n) = 1)) - (Pr(D(Y_n, 1^n) = 1))| < \frac{1}{p(n)}$

5 Secure Contract Framework

In this section we introduce a framework for secure contract negotiation. we begin with several definitions:

- A *clause* is a public set $S = \{s_0, \ldots s_{N-1}\}$ of possible values. We refer to each of these values as *terms*. We assume that Alice and Bob can agree on S at the start of the negotiation. Furthermore, there is a defined ordering of the terms, so that s_i is the ith term in the set.
- For each clause S, Alice (Bob) has a set of *constraints* on the acceptability of each of that clause's terms. These constraints are represented by sets A (respectively, B), where $A \subseteq S$ ($B \subseteq S$) and A (B) is the set of all terms for clause S that are acceptable to Alice (Bob).
- A term $x \in S$ is *acceptable* iff $x \in (A \cap B)$.
- A clause is *satisfiable* iff $A \cap B \neq \emptyset$, i.e., there is a term for the clause that is acceptable to both Alice and Bob.
- A *negotiation* is a sequence of clauses $S_0, \ldots, S_{k-1}$. In this paper, we assume that these clauses are independent (i.e., that the acceptability of one clause does not depend on the outcome of another clause). We briefly discuss how to extend our protocols for dependent clauses in Section 8.3. A negotiation is *satisfiable* iff each clause is satisfiable.
- A *contract* for a negotiation is a sequence of terms $x_0, \ldots, x_{k-1}$ (where $x_i \in S_i$). A contract is *valid* if each term is acceptable to both parties. A valid contract is *efficient* if no term in it is replaceable by another term that is better for both Alice and Bob (according to their respective private valuation functions).

Example: Suppose Alice and Bob are entities that jointly manufacture some type of device, and furthermore they must negotiate where to manufacture the devices. The clause could take the form $S = \{$London, New York, Chicago, Tokyo, Paris, Ottawa$\}$. Now suppose Alice's constraints are $A = \{$London, New York, Paris, Ottawa$\}$ and Bob's constraints are $B = \{$London, Tokyo, Paris, Ottawa$\}$. The set of acceptable terms are those in $A \cap B = \{$London, Paris, Ottawa$\}$.

We now outline the framework for our protocols. Protocols for computing the satisfiability of a clause and an acceptable term for the clause are given in the next section. However, this needs to be extended to the contract level, because the individual results for each clause cannot be revealed when the negotiation is not satisfiable. Given a protocol that evaluates whether or not a clause is satisfiable in a split manner, it is possible to determine if the contract is satisfiable. This is obvious since a contract is satisfiable iff all clauses are satisfiable, which can be computed easily by computing the AND of many split Boolean values using *Secure Circuit Evaluation* [28]. A key observation is that if a contract is satisfiable, then to find a valid and fair contract one can find the individual fair clause agreements independently. Thus given secure protocols for determining whether a clause is satisfiable and a protocol for determining an acceptable fair term for a satisfiable clause, it is possible to compute these same values at the contract level.

6 Secure Contract Term Protocols

In this section we propose private protocols for computing: i) the satisfiability of a clause(yes/no) and ii) a fair agreement for satisfiable clauses (recall that fair means that the term is chosen uniformly from all the set of acceptable terms and that neither party has control over the outcome). We postpone discussion of protocols for computing efficient agreements until Section 7. We now define some notation for our protocols. We assume that Alice and Bob are negotiating a specific clause with N terms. We define Alice's (Bob's) acceptability for term i to be a boolean value a_i (b_i).

6.1 Determining satisfiability

A clause is satisfiable iff $\bigvee_{i=0}^{N-1} a_i \wedge b_i$ is true. Clearly this satisfiability predicate be computed (with *Secure Circuit Evaluation*) with $O(N)$ communication and $O(1)$ rounds.

6.2 Computing a fair acceptable term

In this section we introduce a protocol for computing a fair acceptable term for a clause that is known to be satisfiable. The protocol can be described as follows:
Protocol Description:

Input: Alice has a set of binary inputs $a_0, \ldots, a_{N-1}$ and likewise Bob has a set of inputs: $b_0, \ldots, b_{N-1}$. Furthermore it is known that $\exists i \in [0, N)$ such that $a_i \wedge b_i$ is true.

Output: An index j such that $a_j \wedge b_j$ is true, and if there are many such indices, then neither party should be able to control which index is chosen (by modifying their inputs).

Figure 1 describes our protocol for computing a fair acceptable term. However, we also discuss three elements about the protocol's difficulty including: i) we show that semi-honest chosen OT reduces to the above mentioned problem, ii) we discuss a solution using circuit simulation for this problem, and iii) we show a false start for the problem.

A reduction from semi-honest OT:

Suppose Bob is choosing 1 out of N items (item i) from Alice's list of binary values $v_0, \ldots, v_{N-1}$. Alice and Bob define a list of $2N$ values. Alice creates a list where item a_{2j+v_j} is true and a_{2j+1-v_j} is false for all $j \in [0, N)$. Bob creates a similar list, but sets only values b_{2i} and b_{2i+1} to true (and all other values are set to false). Alice and Bob engage in the above mentioned protocol. Clearly from the returned value, Bob can deduce the value of Alice's bit v_i.

Using circuit simulation

In order to make the term fair, the participants could each input a random permutation into the circuit that would compose the permutations and then permute the list with the composed permutation. The circuit would then choose the first value in the list that was an agreement. This would be fair because if at least one party chose a random permutation than the composed permutation would also be random (making the first acceptable item a fair choice). However, this would require at least $O(N \log N)$ inputs into the circuit (and thus this many 1-out-of-2 OTs) as this is the minimum number of bits to represent a permutation. Also, the circuit would have to perform the permutation, which would involve indirectly accessing a list of size N exactly N times. The direct approach of doing this would require $O(N^2)$ gates. Thus the standard circuit would require at least $O(N \log N)$ OTs (also this many modular exponentiations) and $O(N^2)$ communication. The protocol we outline below requires $O(N)$ modular exponentiations and has $O(N)$ communication. Now, it may be possible to reduce the number of bits input into the circuit to $O(N)$ by using a pseudorandom permutation, however this would require the computation of a permutation, which would be a difficult circuit to construct.

Some false starts for Sub-linear communication

It would be desirable for a protocol to have sub-linear (in terms of the number of possible terms) communication. A possible strategy for this is to use a randomized approach. This solution works well if it is known that the sets have a "substantial" intersection, but all that is known is that there is at least one item that is in the intersection. Furthermore, the participants do not want to leak additional information about their own sets, including information about their mutual intersection. And thus any probabilistic strategy must behave as if there is only a single item in the intersection, and such a protocol would not have sub-linear communication. As a final note if it the contract is to be fair and efficient then there is a communication complexity of $\Omega(N)$ for finding such a contract (we prove this in Section 7).

Input: Alice has a set of binary inputs $a_0, \dots, a_{N-1}$ and likewise Bob has a set of inputs: $b_0, \dots, b_{N-1}$. Furthermore it is known that $\exists i \in [0, N)$ such that $a_i \wedge b_i$ is true.
Output: An index j such that $a_j \wedge b_j$ is true, and if there are many such indices, then neither party should be able to control which index is chosen.

1. Alice does the following:
   (a) She chooses a semantically-secure homomorphic encryption function E_A (with modulus M_A) and publishes its public keys and public parameters.
   (b) For each item a_i in the list $a_0, \dots, a_{N-1}$, she creates a value: $\alpha_i \leftarrow E_A(a_i)$. She sends these values to Bob.
2. Bob does the following:
   (a) He chooses a semantically-secure homomorphic encryption function E_B (with modulus M_B) and publishes the public keys and the public parameters.
   (b) For each i from 0 to $N-1$, Bob chooses/computes:
      i. A random value r_i chosen uniformly from $\{0, 1\}$.
      ii. If $b_i = 0$, then he sets $\beta_i \leftarrow E_A(0)$, and otherwise he sets it to $\beta_i \leftarrow \alpha_i * E_A(0)$.
      iii. if $r_i = 0$, then $\gamma_i = \beta_i$, and otherwise $\gamma_i = ((\beta_i * E_A(M_A - 1))^{M_A - 1})$.
      iv. $\delta_i[0] \leftarrow E_B(r_i)$ and $\delta_i[1] \leftarrow E_B(1 - r_i)$

   Bob forms ordered triples $(\gamma_i, \delta_i[0], \delta_i[1])$ and randomly permutes all of the tuples (storing the permutation Π_B), and he sends the permuted list of ordered triples to Alice.
3. Alice permutes the triples using a random permutation Π' and then for each triple in the permuted list $(\gamma_i, \delta_i[0], \delta_i[1])$ (note that these i values are not the same ones that Bob sent, but are the new values in the permuted list) she computes/chooses:
   (a) $\zeta_i \leftarrow \delta_i[D_A(\gamma_i)] * E_B(0)$
   (b) $\eta_i \leftarrow \zeta_i * (\eta_{i-1})^2$ (if $i = 0$, then she sets it to ζ_0).
   (c) She chooses a random q_i uniformly from $\mathbb{Z}^*_{M_B}$.
   (d) $\theta_i \leftarrow (\eta_i * E_B(-1))^{q_i}$

   Alice permutes the θ values using another random permutation Π'' and she computes the permutation $\Pi_A = \Pi''\Pi'$. She sends the permuted θ values along with the permutation Π_A
4. Bob decrypts the values with D_B and finds the value that decrypts to 0; he then finds the original index of this value by inverting the permutation and he announces this index.

Fig. 1: Protocol FIND-AGREEMENT

Proof of Correctness:

Before discussing the security of the above protocol, we show that the protocol is correct in that it computes an agreement. It is easy to verify that the permutations do not effect the result as they are reversed in the opposite order that they were used, and thus our correctness analysis ignores the permutations. We consider a specific term with respective acceptability for Alice and Bob as a_i and b_i (we use c_i to denote $a_i \wedge b_i$). We now trace the protocol describing each variable:

1. The value α_i is $E_A(a_i)$.
2. The value β_i is $E_A(c_i)$.

3. It is easy to verify that the value γ_i is $E_A(c_i \oplus r_i)$ (where $\oplus$ denotes exclusive-or).
4. The value $\delta_i[0]$ is $E_B(r_i)$ and the value $\delta_i[1]$ is $E_B(1 - r_i)$
5. Now, ζ_i is $\delta_i[0]$ when $c_i = r_i$ and is $\delta_i[1]$ otherwise. This implies that $\zeta_i = E_B(c_i)$.
6. Let $\hat{\imath}$ be the first index where $\zeta_{\hat{\imath}}$ is $E_B(1)$. For $i < \hat{\imath}$, the value η_i will be $E_B(0)$. Furthermore, the value $\eta_{\hat{\imath}}$ will be $E_B(1)$. However, for $i > \hat{\imath}$ the value η_i will be something other than $E_B(1)$, because $\eta_i = \zeta_i + {\eta_{i-1}}^2$.
7. If $\eta_i = E_B(x_i)$, the value θ_i will be $E_B(q_i(x_i - 1))$, this value will be $E_B(0)$ only when $x_i = 1$, which will only happen at $i = \hat{\imath}$.

$\square$

Proof of Security (semi-honest model)

There are two parts to proving that this protocol is secure: i) that Alice (or Bob) does not learn additional information about indices that are not the output index, and ii) since we consider the permutations to be inputs into the protocol, we must show that that a party cannot choose its permutation to affect the outcome. Since Alice and Bob's roles are not symmetrical in the protocol, we must prove security for both cases.

Alice

We introduce an algorithm S_B that takes Alice's inputs and creates a transcript that is computationally indistinguishable from Alice's view in the real model. This algorithm is shown in Figure 2.

Input: Alice sees $a_0, \ldots, a_{N-1}, E_A, D_A, E_B$ and she sees the output index j.
Output: Alice must generate values indistinguishable from Bob's values in step 2; these values are triples of the form $(\gamma_i, \delta_i[0], \delta_i[1])$

1. Alice generates a sequence of values $\hat{b}_0, \ldots, \hat{b}_{N-1}$ where where $\hat{b}_i$ is chosen uniformly from $\{0, 1\}$ if $i \neq j$ and is 1 if $i = j$.
2. Alice generates a sequence of random bits $r_0, \ldots, r_{N-1}$ chosen uniformly from $\{0, 1\}$.
3. Alice creates tuples of the from $(E_A(r_i \oplus (a_i \wedge \hat{b}_i), E_B(r_i), E_B(\neg r_i))$.
4. Alice permutes the items using a random permutation and then outputs these tuples.

Fig. 2: Algorithm S_B

Lemma 1. *In the semi-honest model, S_B is computationally indistinguishable from Alice's view from running FIND-AGREEMENT.*

Proof: Since E_B is semantically-secure, the second and third elements of the tuple are indistinguishable from the real execution. To show that the first item is computationally indistinguishable, we must show two things: i) that the decrypted values are indistinguishable (since Alice knows D_A), and ii) that from

Alice's previous information that she created in Step 1 she cannot distinguish the values.

To show (i), the decrypted values from S_B are chosen uniformly from $\{0, 1\}$. In the real execution the values are $E_A(c_i \oplus r_i)$, where r_i is chosen uniformly from $\{0, 1\}$. Thus, the sequences are indistinguishable.

Part (ii) follows from the statement that Bob performs at least one multiplication on each item or he generates the values himself. By the properties of semantically secure homomorphic encryption, this implies that these values are indistinguishable. □

Lemma 2. *In the semi-honest model, Alice cannot control which term is chosen by selecting her permutation in the protocol FIND-AGREEMENT.*

Proof: The composition of two permutations, with at least one being random, is random. Thus, when Bob randomly permutes the tuples in Step 2, Alice cannot permute them in a way that benefits her, as she does not know Bob's permutation. Thus, when she computes the θ values the permutation is random and the term is chosen fairly. □

<u>**Bob**</u>

We introduce an algorithm S_A that takes Bob's inputs and creates a transcript that is computationally indistinguishable from Alice's view in the real model. This algorithm is shown in Figure 3.

Lemma 3. *In the semi-honest model, S_A is computationally indistinguishable from Bob's view from running FIND-AGREEMENT.*

Proof: Since E_A is semantically-secure, the values $E_A(\hat{a}_0), \ldots, E_A(\hat{a}_{N-1})$ are indistinguishable from the real execution and the permutation $\hat{\Pi}$ is also indistinguishable. To show that the the values $\hat{\theta}_0, \ldots, \hat{\theta}_{N-1}$ are computationally indistinguishable from the real execution, we must show two things: i) that the decrypted values are indistinguishable (since Bob knows D_B), and ii) that from his computations from Step 2, he cannot distinguish the values.

To show (i), the values in S_A are $N - 1$ random values and a single 0 value where the 0 is placed randomly. And since in Step 3.d of the protocol, Alice multiplies the values by a random value and then permutes the items these values are indistinguishable.

To show (ii), all that needs to be shown is that Alice performs a multiplication on each item, and this is clearly done in Step 3.a of the protocol. □.

Lemma 4. *In the semi-honest model, Bob cannot control which term is chosen by selecting his permutation in the protocol FIND-AGREEMENT.*

Proof: The composition of two permutations, with at least one being random, is random. Thus when Alice permutes the list with Π' the values are randomly permuted, and when the first agreement is chosen from this list, it is fairly chosen. □

Input: Bob sees $b_0, \ldots, b_{N-1}, E_B, D_B, E_A$ and he sees the output index j.
Output: Bob must generate values indistinguishable from Alice's values in steps 1 and 3; these values include: $E_A(a_0), \ldots, E_A(a_{N-1}), \theta_0, \ldots, \theta_{N-1}$ and Π.

1. Bob generates a sequence of values $\hat{a}_0, \ldots, \hat{a}_{N-1}$ where where $\hat{a}_i$ is chosen uniformly from $\{0, 1\}$ if $i \neq j$ and is 1 if $i = j$.
2. Bob generates a list of N items $\bar{\theta}_0, \ldots, \bar{\theta}_{N-1}$ where the jth value is 0 and all other values are chosen uniformly from $\mathbb{Z}^*_{M_B}$. He then creates a random permutation $\hat{\Pi}$ and permutes the values. Call this permuted list $\hat{\theta}_0, \ldots, \hat{\theta}_{N-1}$.
3. Bob outputs $E_A(\hat{a}_0), \ldots, E_A(\hat{a}_{N-1}), \hat{\theta}_0, \ldots, \hat{\theta}_{N-1}$ and $\hat{\Pi}$.

Fig. 3: Algorithm S_A

7 Expressing preferences

It is of course unrealistic to assume that Alice and Bob have sets of acceptable states that are all equally desirable. There are many terms for a clause that are a win-win situation for Alice and Bob (i.e., both prefer a specific term), however the random selection provided by FIND-AGREEMENT does not allow the choice of contracts that are efficient in the sense that both parties may prefer another term. Therefore by efficient we mean Pareto-optimal: Any improvement for Alice must be at the expense of Bob and vice-versa. In this section, we describe an extension that allows Alice and Bob to make preference choices through arbitrary utility functions that assign a desirability score to each term. We then filter out all terms that are not Pareto-optimal.

Let $U_A(x)$ (respectively, $U_B(x)$) denote Alice's (Bob's) utility for term x. In this section we introduce a filtering protocol FILTER, that filters out inefficient solutions. We assume that any terms that are deemed unacceptable to a party have utility of 0 for that party, and we assume that all acceptable terms have unique utility (i.e, there are no ties). This last constraint is reasonable since if two terms have equal desirability, then the parties can easily just assign them unique utilities in a random order.
Example: Returning to our example where S = {London, New York, Chicago, Tokyo, Paris, Ottawa}, A = {London, New York, Paris, Ottawa} and B = {London, Tokyo, Paris, Ottawa}. Suppose Alice sets her utilities to {London(3), New York(4), Chicago(0), Tokyo(0), Paris(1), Ottawa(2)}, and Bob sets his utilities to {London(3), New York(0), Chicago(0), Tokyo(1), Paris(4), Ottawa(2)}. Recall that the original list of acceptable terms with utilities is {London(3,3), Paris(1,4), Ottawa(2,2)}. In this case Ottawa is an inefficient solution for this negotiation, because both parties prefer London to it.

It suffices to give a protocol for marking the terms of S that are inefficient. We do this by computing a value between Alice and Bob that is XOR-split (i.e., each party has a value, and the exclusive-or of their values is equal to the predicate "term is efficient"). It is a natural extension of the FIND-AGREEMENT protocol to utilize such values and we omit the details. We omit a detailed proof of security, as this is a natural extension to the proofs outlined before. This filtering process is described in Figure 4.

Input: Alice has binary values $a_0, \ldots, a_{N-1}$, a set of integer utilities $A_0, \ldots, A_{N-1}$, a homomorphic encryption schemes E_A (where the modulus is M_A) and D_A. Bob also has a list of binary values $b_0, \ldots, b_{N-1}$, a set of integer utilities $B_0, \ldots, B_{N-1}$, and has E_A. It is also known that there is a term where $a_i \wedge b_i = 1$.
Output: Alice has binary values $\bar{a}_0, \ldots, \bar{a}_{N-1}$ and Bob has $\bar{b}_0, \ldots, \bar{b}_{N-1}$ where $\bar{a}_i \oplus \bar{b}_i = a_i \wedge b_i$ and the utility (A_i, B_i) is not dominated by another term. Furthermore, this list can be in any order.

1. Alice sends Bob $E_A(A_0), \ldots, E_A(A_{N-1})$.
2. For each i from 0 to $N-1$, Bob does the following:
   (a) Bob chooses a random values r_i in $\mathbb{Z}_{M_A}$.
   (b) If $b_i = 1$, then Bob computes $\alpha_i = E_A(a_i) * E_A(-r_i)$. And if $b_i = 0$, then Bob computes $\alpha_i = E_A(-r_i)$

   Bob sorts the α values in descending order according to his utility function. He sends these "sorted" values to Alice.
3. Alice and Bob engage in a Scrambled Circuit Evaluation that computes the max of the first i items and then if the $(i+1)$st item is smaller than this max it replaces it by 0, otherwise it replaces it with 1. This is done in a XOR-split fashion. Clearly, this circuit can be done with $O(N)$ comparison circuits. One practical matter is the the comparison circuits must be able to compare ρ bits, where ρ is the security parameter for a homomorphic scheme (which has a substantial number of bits). However, the techniques in [7] can be used to reduce the number of bits used by the comparison circuit substantially.

Fig. 4: Protocol FILTER

As a final note, we prove that finding a fair and efficient term has a communication complexity of $\Omega(N)$. We do this by showing a reduction from Set Disjointness (which has a lower bound of $\Omega(N)$ [18]. We now give a sketch of this proof:

Suppose Alice has a set A and Bob has a set B. Alice and Bob define another item (call it c) and both include it in their sets. They assign utilities to all items in their sets randomly, with the condition that the utility of c has to be lower than the utilities of all other items in their sets. They engage in a protocol to find a fair and efficient item. If the item is c, then the sets are disjoint and if the item is not c then the sets are not disjoint. $\square$

8 Extensions

In this section we outline three extensions (due to page constraints we omit many details). In section 8.1 we discuss how to have some types of interactive negotiation. In section 8.2 we discuss how to make our protocol's communication proportional to $O(|A| + |B|)$ (which could be more efficient than our previous solution). Finally, in section 8.3 we outline how to handle dependent contract terms.

8.1 Interactive Negotiations

Consider what happens when the negotiators run the protocol and the output is that no agreement is possible. If the parties stopped here then this system may not be very useful for them. We now outline some strategies that will help them negotiate contracts in a more "interactive" fashion. One of the problems with these approaches is that they allow the entities to perform some level of probing. Some of these strategies require changes to our protocols, but we leave the details for the final version of the paper: i) the parties could change their values and run the protocol again, ii) the parties could make several acceptability sets (in some order of acceptability) and run a protocol that uses all of these sets as a batch, and iii) the protocols could give some feedback to the users. Some possible types of feedback include: what are the clauses without an agreement, if the number of clauses without an agreement is below some threshold than what are the clauses, or based on thresholds the protocols could output some metric as to how far away the parties are from an agreement.

8.2 Efficient Communication

The protocols outlined before our not particularly efficient if Alice and Bob's acceptability sets are much smaller than N. It would be desirable to have protocols with communication proportional to $|A| + |B|$. The downside to such a system is that it reveals "some" additional information, but we believe there are situations where such values are acceptable to leak. Our protocols can be modified to support such clauses, through usage of the protocols in [6].

8.3 Dependent Contract Terms

In this section we briefly outline an extension to our framework for dependent clauses. Two clauses are *dependent* if the value of one clause affects the acceptability set of another clause. For example, the location of a contract might effect which tasks a company is willing/capable to do. Another issue with dependency is if the dependency relationship is known globally or if it must be hidden. Here we assume that information about which clauses are dependent is public.

We now present a more formal definition of two-clause dependency (which can easily be generalized to n-clause dependency). Alice views clause C_2 as *dependent* on clause C_1 if the acceptability set for C_2 (call it A_2) is a function of the term value chosen for C_1. Any contract with dependent clauses can be handled with our framework by taking every group of dependent clauses $C_1, \ldots, C_k$ and making a "super"-clause to represent all of the clauses. The set of states for this "super"-clause would be the k-tuples in the set $C_1 \times \ldots \times C_k$.

9 Summary

In this paper we define protocols for negotiating a contract between two entities without revealing their constraints for the contract. There are two essential issues that need to be addressed: i) is there an agreement for a contract and ii) if there is an agreement, then what is a valid, fair, and efficient contract. To provide efficiency we propose assigning utilities to terms and then filtering out inefficient solutions. To provide fairness the protocols choose a random efficient term in such a way that neither party has control over the choice of the term; the protocol for achieving fairness is the centerpiece of this exposition. Furthermore, the protocols can be extended to handle contracts with publicly-known inter-clause dependencies. This is a first step in the area of secure contract negotiation; possible future work includes: i) protocols for dependent clauses that are better than the generic equivalents, ii) protocols for specific terms that are more efficient than the generic protocols presented in this paper, iii) extending the framework to more than two parties, iv) extending the protocols to a model of adversary besides semi-honest, and v) extending the framework to allow multiple negotiations with inter-contract dependencies.

Acknowledgments

The authors would like to thank the anonymous reviewers for their useful comments and suggestions.

References

1. Michael Ben-Or and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10. ACM Press, 1988.
2. R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.
3. Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 494–503. ACM Press, 2002.
4. David Chaum, Claude Crépeau, and Ivan Damgard. Multiparty unconditionally secure protocols. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 11–19. ACM Press, 1988.
5. Ivan Damgård and Mads Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In *4th International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2001*, LNCS 1992, pages 119–136, 2001.
6. M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *International Conference on the Theory and Application of Cryptographic Techniques, EUROCRYPT 04*, 2004.
7. K. Frikken and M. Atallah. Privacy preserving route planning. In *To appear in Proceeding of the ACM workshop on Privacy in the Electronic Society.* ACM Press, 2004.
8. O. Goldreich. *Foundations of Cryptography: Volume I Basic Tools.* Cambridge University Press, 2001.
9. O. Goldreich. *Foundations of Cryptography: Volume II Basic Application.* Cambridge University Press, 2004.

10. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proceedings of the nineteenth annual ACM conference on Theory of computing*, pages 218–229. ACM Press, 1987.
11. Oded Goldreich. Secure multi-party computation. Working Draft, 2000.
12. Oded Goldreich. Cryptography and cryptographic protocols. *Distrib. Comput.*, 16(2-3):177–199, 2003.
13. Shafi Goldwasser. Multi party computations: past and present. In *Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, pages 1–6. ACM Press, 1997.
14. Guido Governatori, Arthur H.M. ter Hofstede, and Phillipa Oaks. Defeasible logic for automated negotiation. In P. Swatman and P.M. Swatman, editors, *Proceedings of CollECTeR*. Deakin University, 2000. Published on CD.
15. Benjamin N. Grosof, Yannis Labrou, and Hoi Y. Chan. A declarative approach to business rules in contracts: courteous logic programs in XML. In *ACM Conference on Electronic Commerce*, pages 68–77, 1999.
16. R. Impagliazzo and S. Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 44–61. ACM Press, 1989.
17. J. Katz and R. Ostrovsky. Round optimal secure two-party computation. In *CRYPTO 04*, 2004.
18. E. Kushilevitz and N. Nisan. *Communication Complexity.* Cambridge University Press, 1997.
19. D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella. Fairplay - a secure two-party computation system. In *Proceedings of Usenix Security*, 2004.
20. Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 245–254. ACM Press, 1999.
21. Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 448–457. Society for Industrial and Applied Mathematics, 2001.
22. T. Okamoto, S. Uchiyama, and E. Fujisaki. Epoc: Efficient probabilistic public-key encryption, 1998.
23. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International Conference on the Theory and Application of Cryptographic Techniques, EUROCRYPT 99*, LNCS 1592, pages 223–238, 1999.
24. Bruce Schneier. *Applied Cryptography – Protocols, algorithms, and souce code in C.* John Wiley & Sons, Inc., 1996.
25. R. Smith and J. Shao. Preserving privacy when preference searching in e-commerce. In *Proceeding of the ACM workshop on Privacy in the Electronic Society*, pages 101–110. ACM Press, 2003.
26. Michael Strbel. Intention and agreement spaces - a formalism.
27. A.C Yao. Protocols for secure computation. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 160–164, 1982.
28. A.C Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.