Desk-mates (Stable Matching) with Privacy of Preferences, and a new Distributed CSP Framework

Marius-Călin Silaghi[1] and Markus Zanker[2] and Roman Barták[3]

[1] Florida Institute of Technology, USA
[2] Universität Klagenfurt, Austria
[3] Charles University, Czech Republic
msilaghi@fit.edu, markus@ifit.uni-klu.ac.at, bartak@kti.mff.cuni.cz

Abstract. The desk-mates matcher application solves the need of placing students in pairs of two for working in projects (need that is similar to the well known problems of stable matchings or stable roommates). Each of the persons in the previous application has a (hopefully stable) secret preference between every two possible partners. The participants want to find an allocation satisfying their secret preferences and without leaking any of these secret preferences, except for what a participant can infer from the identity of the partner that was recommended to her.
The peculiarities of this problem requires solvers based on old distributed CSP frameworks to use models whose search spaces are higher than those in centralized solvers, with bad effects on efficiency.
We introduce a distributed weighted constraint satisfaction (DisWCSP) framework where the actual constraints are secrets that are not known by any agent. They are defined by a set of functions on some secret inputs from all agents. The solution is also kept secret and each agent learns just the result of applying an agreed function on the solution. The new framework is shown to improve the efficiency ($O(2^{m^3-\log(m)})$ times) in modeling and solving the aforementioned problem with m participants. We show how to extend our previous techniques to solve securely problems modeled with the new formalism, and exemplify with the problem in the title. An applet-based solver is available [Sil04a].

1 Introduction

The desk-mates matcher application groups a set of students in stable working teams of two, such that whenever one person wants to change her partner for a third one, the third one prefers her current partner to the change (similar to stable matchings or stable roommates [IM02]). The students have a secret preference between any pair of potential partners, and between working with any given partner or working alone.

Versions of these problems, without privacy requirements, have been long known and studied. It is an example of constraint satisfaction problem (CSP) [GP02].[1] A CSP is described by a set of variables and a set of constraints on the possible values of those variables. The CSP problem consists in finding assignments for those variables with values from their domains such that all constraints are satisfied. The centralized CSP

[1] Operations research has provided very efficient solutions to some instances without privacy.

techniques require every eventual participant to reveal its preferences (e.g. to a trusted server), to compute the solution. Therefore, they apply only when the participants accept to reveal their preferences to the trusted party.

There exist frameworks and techniques to model and solve distributed CSPs (DisCSPs) with privacy requirements, namely when the domains of the variables are private to agents [YDIK98,MJ00], or when the constraints are private to agents [SSHF00a,Sil03b,SR04].

However, the desk-mates problem seem not to be modeled efficiently (i.e. with a reduced search space) with any of the two known types of distributed CSP frameworks. In this article we propose a new framework for the distributed constraint satisfaction problems. It can model naturally existing distributed constraint satisfaction problems, and also the desk-mates (stable matchings problems). The new framework assumes that the constraints are not known to absolutely any agent but they are computable from secret inputs, by applying public functions on them. These functions use secret inputs provided securely by the different participants. Similarly, the final assignments are secret and each agent can retrieve just the result of applying some agreed function on the secret solution.

We also show how secure multi-party computation techniques that we have recently developed for solving DisCSPs with private constraints can be extended to solve problems described in the new framework. We start introducing formally the CSP problem.

CSP. A *constraint satisfaction problem* (CSP) is defined by three sets: (X, D, C). $X = \{x_1, ..., x_m\}$ is a set of variables and $D = \{D_1, ..., D_m\}$ is a set of finite domains such that x_i can take values only from $D_i = \{v_1^i, ..., v_{d_i}^i\}$. $C = \{\phi_1, ..., \phi_c\}$ is a set of constraints. A constraint ϕ_i limits the legality of each combination of assignments to the variables of an ordered subset X_i of the variables in X, $X_i \subseteq X$. An assignment is a pair $\langle x_i, v_k^i \rangle$ meaning that the variable x_i is assigned the value v_k^i.

A tuple is an ordered set. The projection of a tuple ϵ of assignments over a tuple of variables X_i is denoted $\epsilon_{|X_i}$. A solution of a CSP (X,D,C) is a tuple of assignments, $\epsilon *$, with one assignment for each variable in X such that each $\phi_i \in C$ is satisfied by $\epsilon *_{|X_i}$. The search space of a CSP is the Cartesian product of the domains of its variables.

We consider that a set of participants are the source of such CSPs and one has to find agreements for a solution, from the set of possible alternatives, that satisfies a set of (secret) requirements of the participants. This view suggests a concept of a distributed CSP. Several frameworks were proposed so far for Distributed Constraint Satisfaction [ZM91,CDK91,YSH02a,MJ00]. Some versions consider that each agent owns a constraint of the CSP [ZM91,SGM96]. This constraint could model the private information of the agent [SSHF00a]. Other versions consider that each agent owns the domain of a variable while the constraints are shared [YDIK98]. The secret domains can also model some private constraints of the agent.

None of the two approaches, namely private variables or private domains, can model efficiently the stable matching problems. This is because the private data of these problems does not *directly* constrain the allocation of the natural shared resources (the matching). An indirect relation exist with such a constraint. Redundant variables would need to be introduced in the system, modeling the secret preferences, but reducing efficiency. A new framework will be introduced in this article to avoid these redundant variables.

2 The Desk-mates Matcher Application

In some of our classes students are grouped in teams of two, for solving laboratory excercises as well as for working on projects. It is desirable for these teams to be stable for the duration of the project. Otherwise discontinuities and changes may reduce the efficiency of the learning process. Some students insist to work alone, and it is typically difficult for students to refuse other's offers of partnership. In fact students sometimes prefer to keep private their preferences between colleagues, to avoid hurting others. We decided that it is needed to provide students with a support in solving this situations. We therefore built a web-application that insures their privacy using cryptographic solvers of distributed constraint satisfaction problems, as proposed in this paper.

Our web-application works as follows. An organizer of the computation, e.g. an instructor or a student, uses the web form at [Sil04a] to generate (for the included JAVA applets) parameters that are customized for the computation at hand. This process requires the organizer to input the size of the class, the names of the students, and a cryptographic public Paillier key provided by each student. Students can generate Paillier key pairs using the corresponding applet linked from the form, and keep the secret keys while handing the private ones to the organizer.

When the customized problem description is generated, a website is automatically built and provided for this problem instance. The organizer is offered an opportunity to email its URL to the students. The organizer can also specify which algorithm to be used for the computation.

Each student browses the received URL, and downloads the applet with customized parameters. The browser can verify the integrity of the applet. Each student provides the applet with his secret key, and inputs his secret preferences. Then he launches his applet into the computation. The applets retrieve each-other's network IP number and port by using a directory server installed on the same host as the web-application. The applets solve the problem securely, and display for each student only the name of her/his partner.

The Distributed Configurator Application Our approach can also be applied to the problem of distributed configuration of products based on components from several providers, with secret configuration requirements.

3 Background

Our techniques here apply only to problems whose constraints and outputs can be represented as first order logic expressions, or as arithmetic circuits on inputs. Actually, we propose a procedure to translate first order logic definitions of constraints/outputs into arithmetic circuits. In the following we introduce arithmetic circuits and a short overview of the literature and techniques that made them relevant.

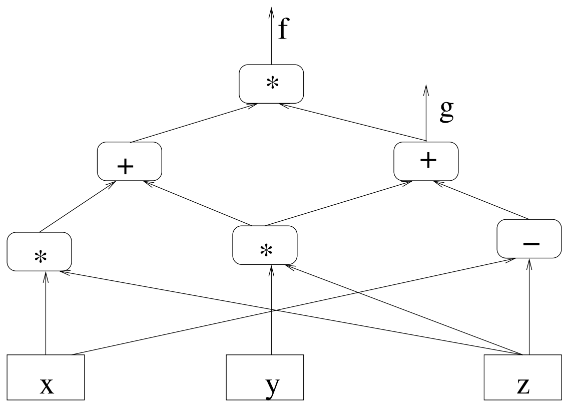

Fig. 1. An arithmetic circuit, $g = yz + (x - z)$ and $f{=}(xz + yz)g$. Each input can be the secret of some participant. The output may not be revealed to all participants. All intermediary values remain secret to everybody.

3.1 Secure Arithmetic Circuit Evaluation

Secure multi-party computations can simulate any arithmetic circuit [BOGW88] or boolean circuit [Kil88,Gol04] evaluation. An *arithmetic circuit* can be intuitively imagined as a directed graph without cycles where each node is described either by an addition/subtraction or by a multiplication operator (see Figure 1). Each leaf is a constant. In a secure arithmetic circuit evaluation, a set of participants perform the operations of an arithmetic circuit over some inputs, each input being either public or an (encrypted/shared) secret of one of them. The result of the arithmetic circuit are the values of some predefined nodes. The protocol can be designed to reveal the result to only a subset of the agents, while none of them learns anything about intermediary values. One says that the multi-party computation *simulates* the evaluation of the arithmetic circuit. A *boolean circuit* is similar, just that the leafs are boolean truth values, false or true, often represented as 0 and 1. The rest of the nodes are boolean operators like AND or XOR. A function does not have to be represented in this form to be solvable using general secure arithmetic circuit evaluation. It only needs to have such an equivalent representation. For example, the operation $\sum_{i=B}^{E} f(i)$ is an arithmetic circuit if B and E are public constants and $f(i)$ is an arithmetic circuit. The same is true about $\prod_{i=B}^{E} f(i)$. Such constructs are useful when designing arithmetic circuits.

There must be some machinery to compute the result of the circuit from the inputs. However, existing techniques allow for the secret inputs not to be revealed to this machinery. Namely the machinery works only with encrypted secrets that it cannot decrypt (i.e. using Shamir's secret sharing [Sha79], detailed later).

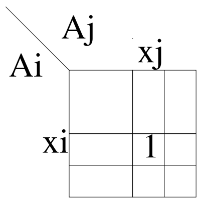

Fig. 2. A constraint between x_i and x_j for the desk-mates problems. An element is feasible, value '1', if the correponding pairs (A_i,A_{x_i}) and (A_j,A_{x_j}) are allowed in a solution given the preferences of $A_i, A_j, A_{x_i}, A_{x_j}$.

4 Distributed CSPs with constraints secret to everybody

In this article we redefine the distributed CSP framework, aiming to model efficiently (i.e. with a reduced search space) the distribution of some famous CSP problems, namely the stable machings problems (e.g. the desk-mates problem).

Desk-mates The desk-mates problem consists in placing a set of persons $A = \{A_1, ..., A_m\}$ in teams of two (or two-seats desks), such that if any person A_i prefers a person A_j to the desk-mate selected for her, then A_j prefers her current desk-mate to A_i.

A way of modeling the desk-mates problem as a CSP is to have one variable x_i for each person A_i specifying the index of the desk-mate assigned to her by the solution, or specifying i, the index of A_i itself, if she remains alone. The constraints are obtained by preprocessing the input from participants about their preferences. The fact that a person A_i prefers A_u to A_v is specified by the first order logic predicate $P_{A_i}(u, v)$. There is a constraint ϕ^{ij} between every pair of distinct variables x_i and x_j. In first order logic notation, the constraint between each two variables x_i and x_j is:

$$\forall x_i, x_j : \phi^{ij}(x_i, x_j) \overset{\text{def}}{=} (P_{A_i}(x_j, x_i) \Rightarrow P_{A_{x_j}}(j, i)) \wedge (P_{A_j}(x_i, x_j) \Rightarrow P_{A_{x_i}}(i, j)) \wedge ((x_i = j) \Leftrightarrow (x_j = i)) \quad (1)$$

Read: *For each pair of participants A_i, A_j, (and corresponding variables x_i and x_j) there is a constraint ϕ^{ij} that allows a pair of assignments to these variables only if:*

- *the fact that A_i prefers the participant assigned to A_j (A_{x_j}) to her own match A_{x_i} implies that:*
  the agent assigned by these assignments to A_j (A_{x_j}), prefers the agent A_j to the agent A_i.
- *the fact that A_j prefers the participant assigned to A_i (A_{x_i}) to her own match A_{x_j} implies that*
  the agent assigned by these assignments to A_i (A_{x_i}), prefers the agent A_i to the agent A_j.
- *A_j is the match of A_i only if A_i is the match of A_j.*

Note that this model subsumes the constraints: $\forall i, j : x_i \neq x_j$. The main complication with this kind of CSPs is that the constraints are functions of secrets that cannot be easily elicited from the participants. Distributed CSP frameworks are meant to address such problems.

Modeling the desk-mates problem with DisCSPs with secret constraints that are known to some agents. One can model the desk-mates problem with secret constraints known to some agents [ZM91,SSHF00b] by choosing as variables, $x_1, ..., x_m$, the index of the partner associated to each agent (that has to be computed) and using one additional boolean variable for each secret preference, $P_{A_i}(u, v)$. The total number of boolean variables is m^3, m^2 of them being actually fixed by public constraints (e.g. $P_{A_i}(u, u) = 0$). However, also taking into account the variables $x_1, ..., x_m$, the total search space becomes $O(m^m 2^{m^3})$. This is $O(2^{m^3})$ times worse than the centralized CSP formalization whose search space is only $O(m^m)$.

We propose now a distributed constraint satisfaction framework that allows to model these problems with the same search space size as the CSP framework, $O(m^m)$.

4.1 Redefining the Distributed Constraint Satisfaction Framework

In the previous part of this section we have exemplified CSP models for the stable matchings problem. We have seen that it is difficult to model efficiently these problems using existing private variable-, or private constraint- oriented distributed constraint satisfaction frameworks.

Let us propose a framework for modeling distributed CSPs, where a constraint is not (necessarily) a secret known to an agent, or public, but can also be a secret unknown to all agents.

Any distributed problem is essentially (in our view) described by a set of inputs and expected outputs from/to each participant. A distributed CSP is a specialization in the sense that the inputs are used to specify constraints/domains of a CSP, and the outputs are derived from the solution of that CSP. As shown elsewhere, sometime the inputs are also needed (in combination with the solution) to provide meaningful outputs [Sil04b].

Definition 1. *A Distributed CSP (DisCSP) is defined by six sets (A, X, D, C, I, O) and an algebraic structure F. $A=\{A_1, ..., A_n\}$ is a set of agents. X, D, and the solution are defined like for CSPs.*

$I=\{I_1, ..., I_n\}$ is a set of secret inputs. I_i is a tuple of α_i secret inputs (defined on F) from the agent A_i. Each input I_i belongs to F^{α_i}.

Like for CSPs, C is a set of constraints. There may exist a public constraint in C, ϕ_0, defined by a predicate $\phi_0(\epsilon)$ on tuples of assignments ϵ, known to everybody. However, each constraint $\phi_i, i{>}0$, in C is defined as a set of known predicates $\phi_i(\epsilon, I)$ over the secret inputs I, and the tuples ϵ of assignments to all the variables in a set of variables X_i, $X_i \subseteq X$.

$O=\{o_1, ..., o_n\}$ is the set of outputs to the different agents. Let m be the number of variables. $o_i : D_1 \times ... \times D_m \rightarrow F^{\omega_i}$ is a function receiving as parameter a solution and returning ω_i secret outputs (from F) that will be revealed only to the agent A_i.

Theorem 1. *The framework in the Definition 1 can model any distributed constraint satisfaction problems with private constraints [SSHF00b].*

Proof. The new DisCSP framework can be used to model any of the DisCSP problems with constraints private to agents, by defining I_i as the extensional representation of the private constraint of A_i (assuming the simple but sufficient case of one constraint per agent). $\phi_i(\epsilon, I)$ is then given by the corresponding value for ϵ in I_i (true/1 or false/0). The outputs are going to be $o_i(\epsilon) = \epsilon$ for all i. q.e.d.

Theorem 2. *The framework in the Definition 1 can model distributed constraint satisfaction problems with private domains [YDIK98].*

Proof. A private domain of an agent can also be modeled as a private unary constraint, in a DisCSP where each domain is the maximum possible domain for the variable. Then, the Theorem 1 applies. q.e.d.

We do not claim that the new framework is more general than the existing frameworks. It enables us to model naturally and efficiently the desk-mate (stable matchings) problems. One can also model these problems with the old frameworks, but they seem to yield much larger search spaces, and therefore less efficient solutions. Let us now exemplify how this framework can model the new problems.

Modeling the desk-mates problem as a DisCSP. A way of modeling the desk-mates problem as a DisCSP is to have one agent, A_i, and one variable, x_i, for each participant in the problem description. x_i specifies the index of the desk-mate assigned to A_i by the solution, or specifies i if she remains alone. The inputs I_i of each agent are given by the set of preferences $P_{A_i}(u, v)$, specifying whether A_i prefers A_u to A_v, for each u and v. The set F, to which belong the inputs and the outputs, is $\{true, false\}$.

There is a constraint ϕ^{ij} between every pair of variables x_i and x_j, defined as in Equation 1. The output functions are defined as: $o_i(\epsilon) \overset{\text{def}}{=} \epsilon_{|\{x_i\}}$. Namely, each agent learns only the name of her desk-mate. There is a public constraint:

$$\phi_0 \overset{\text{def}}{=} \forall i, j, ((x_i = j) \Leftrightarrow (x_j = i)) \wedge (x_i \neq x_j) \tag{2}$$

4.2 Distributed Weighted Constraint Satisfaction Problems

Definition 2. *A distributed constraint satisfaction problem (DisWCSP) is defined by six sets (A, X, D, C, I, O), and algebraic structure F, and a set of acceptable solution qualities B, that can be often represented as an interval* $[B_1, B_2]$.

- *A=$\{A_1, ..., A_n\}$ is a set of agents.*
- *$X = \{x_1, ..., x_m\}$ is a set of variables and $D = \{D_1, ..., D_m\}$ is a set of finite domains such that x_i can take values only from $D_i = \{v_1^i, ..., v_{d_i}^i\}$. An assignment is a pair $\langle x_i, v_k^i\rangle$ meaning that the variable x_i is assigned the value v_k^i. A tuple is an ordered set.*
- *I=$\{I_1, ..., I_n\}$ is a set of secret inputs. I_i is a tuple of α_i secret inputs (defined on a set F) from the agent A_i. Each input I_i belongs to F^{α_i}.*

- *$C = \{\phi_0, ..., \phi_c\}$ is a set of constraints. A constraint ϕ_i weights the legality of each combination of assignments to the variables of an ordered subset X_i of the variables in X, $X_i \subseteq X$. ϕ_0 is a public constraint defined by a function $\phi_0(\epsilon)$ on tuples of assignments ϵ, known to everybody. Each constraint ϕ_i, $i>0$, in C is defined as a known function $\phi_i(\epsilon, I)$ over the secret inputs I, and the tuples ϵ of assignments to all the variables in a set of variables X_i, $X_i \subseteq X$. $\phi_i(\epsilon, I)$ maps secret inputs and tuples into weights.*
- *The projection of a tuple ϵ of assignments over a tuple of variables X_i is denoted $\epsilon_{|X_i}$. A solution is $\epsilon* = \underset{\epsilon \in D_1 \times ... \times D_n}{\text{argmin}} \sum_{i=0}^{c} \phi_i(\epsilon_{|X_i})$, if $\sum_{i=0}^{c} \phi_i(\epsilon*_{|X_i}) \in [B_1 ... B_2]$.*
- *$O=\{o_1, ..., o_n\}$ is the set of outputs to the different agents. $o_i : I \times D_1 \times ... \times D_m \rightarrow F^{\omega_i}$ is a function receiving as parameter the inputs and a solution, and returning ω_i secret outputs (from F) that will be revealed only to the agent A_i.*

Solvers developed in our previous work require that the functions in sets O and C are input either in first order logic form, or in the form of arithmetic circuits.

The inputs are not revealed to the solving machinery, as it manipulates only encrypted data (in difference from a classic monolithic system with a trusted server).

The public constraint ϕ_0 can be input into the system using a set of constraints $\{\phi_0^1, \phi_0^2, ...\}$, and the tuples of assignments accepted by ϕ_0 can be obtained separately by each agent, when needed, using any systematic search technique that finds all solutions of a CSP, e.g. backtracking or lookahead algorithms (BT, BM, CBJ, FC, MAC, EMAC, etc.).

5 Adapting existing secure solvers to the new DisCSP framework

There exist a large set of algorithms addressing distributed CSPs with privacy of constraints [Sil02,HCN$^+$01,FMW01,WS04,YSH02b,Sil03b]. Note that none of the existing techniques involves propagation, except for a very old variant in [Sil02]. The ones that we succeed to extend to the new framework are:

- Finding the set of all solutions of a distributed constraint problem with secret constraints [HCN$^+$01].
- Finding the first solution in a lexicographic order for a distributed constraint satisfaction problem with secret constraints that are known to some agents [Sil03a].
- Finding a random solution for a DisCSP with secret constraints that are known to some agents [Sil03b].

When a solution is returned to the desk-mates problem, each agent A_i can infer that: *any agent A_k preferred by A_i to her current desk-mate A_j, prefers her current partner to A_i*. If only one solution is returned (picked randomly among the existing solutions), then no other secret preference can be inferred with certainty.

Theorem 3. *The desk-mates problem can have several solutions.*

Proof. Consider a case with three agents, A_1, A_2, A_3 where $P_{A_1}(2,3)$, $P_{A_2}(3,1)$, $P_{A_3}(1,2)$. This is a loop of preferences, and has three stable solutions, the sets of teams $\{(A_1,A_2),(A_3)\}$, $\{(A_2,A_3),(A_1)\}$, $\{(A_3,A_1),(A_2)\}$. Such an example can be constructed out of any similar loop of preferences, of any size.

If there exist several solutions, the agents will prefer not to reveal more then one of them. The remaining solutions would only reveal more secret preferences:

– Typically there is no other fair way, except randomness, to break the tie between several solutions.
– If the single solution that is returned is selected as the first one in some given lexicographic order on the variables and domains of the problem, then additional information is leaked concerning the fact that tuples placed lexicographically before the suggested solution do not satisfy the constraints [Sil03b].

As it follows, if it is known that a certain problem has only one solution, then any technique is acceptable among either:

– Finding and returning all solutions using the technique in [HCN$^+$01], or
– Returning only the first solution (e.g. by sequentially checking each tuple in lexicographical order until a solution is found).

Otherwise, strong privacy requirements make techniques returning a random solution [Sil03b] desirable, despite their potential of having a lower efficiency.

5.1 General Scheme

We will note that the main difference between the new DisCSP framework, and the one with secret constraints that are known to some agents, is that now the constraints need to be computed dynamically from secrets inputs. All the techniques we extend to the new framework contain a component based on Shamir's secret sharing [Sha79]. It is the achievement of this sharing which is most affected by the change in framework. We will start by describing Shamir's secret sharing, its importance in distributed multi-party computations, and them we will introduce our changes.

The secure multi-party simulation of arithmetic circuit evaluation proposed in [BOGW88] exploits Shamir's secret sharing [Sha79]. This sharing is based on the fact that a polynomial $f(x)$ of degree $t{-}1$ with unknown parameters can be reconstructed given the evaluation of f in at least t distinct values of x, using Lagrange interpolation. Absolutely no information is given about the value of $f(0)$ by revealing the valuation of f in any at most $t{-}1$ non-zero values of x. Therefore, in order to share a secret number s to n participants $A_1,...,A_n$, one first selects $t{-}1$ random numbers $a_1,...,a_{t-1}$ that will define the polynomial $f(x) = s+\sum_{i=1}^{t-1}(a_i x^i)$. A distinct non-zero number τ_i is assigned to each participant A_i. The value of the pair $(\tau_i, f(\tau_i))$ is sent over a secure channel (e.g. encrypted) to each participant A_i. This is called a (t,n)-threshold scheme. Once secret numbers are shared with a (t,n)-threshold scheme, evaluation of an arbitrary arithmetic circuit can be performed over the shared secrets, in such a way that all results remain shared secrets with the same security properties (the number of supported colluders, $t-1$) [BOGW88,Yao82]. For [Sha79]'s technique, one knows to perform additions and multiplications when $t \leq (n-1)/2$. Since any $\lfloor n/2 \rfloor$ participants cannot find anything secret by colluding, such a technique is called $\lfloor n/2 \rfloor$-private [BOGW88].

We do not try to encode functions, but only their inputs. All functions (more exactly, arithmetic circuits) that will be computed are public and known by all participants. Their inputs, intermediary values, and outputs are shared secrets. The functions that we are able to compute belong to the class of arithmetic circuits. The techniques computing these function do not reveal any information to anybody, and work by letting agents processing the Shamir shares that they know, and by sharing additional secret values.

The techniques solving DisCSPs with private constraints can be used as a black box, except for the secret constraint sharing. Namely, instead of simply sending encrypted Shamir shares [Sha79] of one's constraint, those shares of the constraints have to be computed from the secret inputs of the agents. We therefore propose to replace the secret sharing/reconstruction steps with simulations of arithmetic circuit evaluation which will compute each $\phi_k(\epsilon, I)$ for each tuple ϵ and for the actual inputs I. This step is called *preprocessing*. Intuitively, preprocessing is the step of computing the encrypted initial parameters of the CSP (i.e. acceptance/feasibility value of a tuple from the point of view of each constraint), out of the provided secret inputs. Preprocessing prepares "the pairs" (y,f(y)) that encode the 0/1 values of the constraints. It is done by evaluating arithmetic circuits.

Similarly, instead of just reconstructing the assignments to variables in a solution ϵ, one will have to design and execute secure computations of the functions $o_k(\epsilon)$. This step is called *post-processing*. Intuitively, post-processing is the step of computing the outputs to be revealed to agents, from the obtained encrypted solution of the DisCSP and secret inputs. We show that in our cases this can also be done using simulations of arithmetic circuit evaluations.

Assume $\mathcal{A}$ is some algorithm using Shamir's secret sharing for securely finding a solution of a distributed CSP (with secret constraints known to some agents). The generic extension of the algorithm $\mathcal{A}$ to solve the DisCSP in the new framework is:

- **Preprocessing:** Share the secrets in I with Shamir's secret sharing scheme. Compute each $\phi_k(\epsilon_{|X_k}, I)$ for each tuple $\epsilon_{|X_k}$ and for the actual inputs I by designing it as an arithmetic circuit and simulating securely its evaluation. The public constraint ϕ_0 can be shared by any agent.
- Run the algorithm $\mathcal{A}$ as a black-box, for finding a solution $\epsilon*$ shared with Shamir's secret sharing scheme, for a DisCSP with parameters (i.e. constraints) shared with Shamir's secret sharing scheme.
- **Post-processing:** Compute each $o_i(\epsilon*)$ by designing it as an arithmetic circuit and simulating securely its evaluation. Reveal the result of $o_i(\epsilon*)$ only to A_i.

5.2 Pre- and post- processing for desk-mate problems

In the remaining part of the article we will prove that it is possible to design the needed preprocessing and post-processing to solve our example of DisCSPs, the desk-mates problem, using the general scheme defined above.

Preprocessing for the desk-mates problem. We assume the same choice of variables, as for the CSP formalization of this problem in Section 4. Let us now show how simple arithmetic circuits can implement the required preprocessing.

Each variable x_i specifies the index of the desk-mate associated to A_i. The input of each agent A_i is a preference value $P_{A_i}(j,k)$ for each ordered pair of agents (A_j, A_k), and specifying whether A_i prefers A_j to A_k. $P_{A_i}(j,k)$=1 if and only if A_i prefers A_j to A_k. Otherwise $P_{A_i}(j,k)$=0. A constraint ϕ^{ij} is defined between each two variables, x_i and x_j. I.e. $\phi^{ij}[u,v]$ is the acceptance value of the pair of matches: $(A_i, A_u), (A_j, A_v)$. One synthesizes $m(m-1)/2$ such constraints:

$$\phi^{i,j}[u,v] = \begin{cases} 0 & \text{when } u = v \\ \\ (1 - P_{A_i}(v,u) * (1 - P_{A_v}(j,i)))* & \\ \qquad (1 - P_{A_j}(u,v) * (1 - P_{A_u}(i,j))) & \text{when } u \neq v \end{cases}$$

The public constraint ϕ_0 (same as in Equation 2) restricts each pair of assignments:

$$\forall \epsilon, \epsilon{=}(\langle x_i, u\rangle, \langle x_j, v\rangle) : \phi_0(\epsilon) \overset{\text{def}}{=} ((u{=}j) \Leftrightarrow (v{=}i)) \wedge (u \neq v)$$

ϕ_0 is known by everybody, and therefore there is no need to compute it with arithmetic circuits. The complexity of this preprocessing is $O(m^4)$ multiplications of secrets (for m^2 binary constraints with m^2 tuples each).

The desk-mates problem does not require any arithmetic circuit evaluation for the post-processing, as each agent A_i learns a value existing in the solution, $o_i(\epsilon) = \epsilon_{|\{x_i\}}$. The participants just reveal to A_i their shares of x_i in the solution.

6 Transforming first order logic in arithmetic circuits

Based on the experience with the examples analyzed so far, we conclude that with the new DisCSP framework it is useful to have a mechanism for automatic translation of first order logic sentences about secrets, into arithmetic circuits.

The main constructs in first order logic whose translation to arithmetic circuits will be given here are: $\forall i \in [1..n]P(i)$, $\exists i \in [1..n]P(i)$, $P \wedge Q$, $P \vee Q$, $\neg P$, $\min_{P(i)}(i)$, and $f = k$, where P and Q are predicates with a true (1) or false (0) value, f is a secret integer in a given interval, [1..n], i is a quantified variable that can take integer values in a given interval, [1..n], and k is a constant. They can also apply to variables and secrets from any finite set of numbers, $S = \{a_1, ..., a_n\}$. $\min_{P(i)} i$ is the function returning the minimum i such that $P(i)$ holds. The equivalent arithmetic circuits are shown in Table 1.

6.1 Complexity

For a problem with size of the search space Θ and c constraints, the number of messages for finding all solutions with secure techniques similar to the one in [HCN+01] is given by $(c-1)\Theta$ multiplications of shared secrets ($n(n{-}1)$ messages for each such multiplication). For the desk-mates problem modeled with the new framework, $\Theta{=}m^m$

First Order Logic Sentence	Equivalent Arithmetic Circuit
P	$\overline{P}$
$\forall i \in [1..n], P(i)$	$\prod_{i=1}^{n} \overline{P(i)}$
$\forall a \in S, P(a)$	$\prod_{i=1}^{n} \overline{P(a_i)}$
$\exists i \in [1..n], P(i)$	$\sum_{i=1}^{n}[\overline{P(i)} \prod_{j=1}^{i-1}(1-\overline{P(j)})]$
$\exists a \in S, P(a)$	$\sum_{i=1}^{n}[\overline{P(a_i)} \prod_{j=1}^{i-1}(1-\overline{P(a_j)})]$
$P \wedge Q$	$\overline{P} * \overline{Q}$
$P \vee Q$	$\overline{P} + (1-\overline{P})\overline{Q}$
$P \Rightarrow Q$	$1-\overline{P}(1-\overline{Q})$
$\neg P$	$1-\overline{P}$
$f = k, (f, k \in [1..n])$ (i.e. test if f equals k, where they are in [1..n])	$\frac{1}{(k-1)!(n-k)!} \prod_{i=1}^{k-1}(f-i) \prod_{i=k+1}^{n}(i-f)$
$f = a_k, (f \in S, k \in [1..n])$ (i.e. test if f equals the k^{th} element of S)	$\frac{\prod_{a_i \in S, i \neq k}(f-a_i)}{\prod_{a_i \in S, i \neq k}(a_k-a_i)}$
$\min\limits_{P(i), i \in [1..n]} i$ (i.e., smallest i s.t. $P(i)$ holds)	$\sum_{i=1}^{n}[i\overline{P(i)} \prod_{j=1}^{i-1}(1-\overline{P(j)})]$

Table 1. Equivalences between first order logic constructs and arithmetic circuits. P and Q are predicates and $\overline{P}$ and $\overline{Q}$ are their equivalent arithmetic circuits. $S = \{a_1, ..., a_n\}$.

and c=1 for the version with a single global constraint, or c=$m^2/2$ for the version with binary constraints. For the case with binary constraints, it yields a complexity of $O(m^{m+2})$. As mentioned before, the preprocessing has complexity $O(m^4)$ multiplications between shared secrets, resulting in a total complexity $O(m^2(m^m + m^2))$.

Solving the same problem with the same algorithm but modeled with the old DisCSP framework with private constraints, $\Theta = m^m 2^{m^3}$ and $c = m$, for one global constraint from each agent. There is no preprocessing, but the total complexity is $O(m^{m+1}2^{m^3})$. The new framework behaves better since $m << 2^{m^3}$. The comparison is similar for other secure algorithms, like MPC-DisCSP1 (see [Sil03b]) whose complexity is given by $O(dm(c{+}m)\Theta)$ multiplications between shared secrets.

Similar improvements can be achieved by applying this new framework to other known problems like incentive auctions and stable marriages problems [Sil04b].

7 Conclusions

DisCSPs [BMM01,SGM96,LV97,Ham99,MR99,ZWW02,BD97,FBKG02,MTSY04] are a very active research area. Privacy has been recently stressed in [MJ00,FMW01,WF02,FMG02,YSH02b] as an important goal in designing algorithms for solving DisCSPs.

In this article we have investigated how versions of old and famous problems, stable matchings problems, can be solved such that the privacy of the participants is guaranteed except for what is leaked by the selected solution. Our approach uses secure simulations of arithmetic circuit evaluations and is therefore robust whenever no majority of the participants colludes to find the secret of the others, and when all agents follow the protocol.

We note that the desk-mates problems cannot be efficiently modeled (at least not in an obvious way) with existing distributed constraint satisfaction frameworks. We have therefore introduced a new distributed constraint satisfaction framework that can model such problems with the same search space size as the classic centralized CSP models. We have shown how some techniques for the existing frameworks can be adapted to problems modeled with the new DisCSPs, and we exemplify the model with the desk-mates problems. For m participants in the desk-mates problem, the size of the search space in the DisCSP model achieved with the new framework is $O(m^m)$ while the previous framework with private constraints yields DisCSP instances with a size of the search space of $O(m^m 2^{m^3})$. In existing secure algorithms for solving DisCSPs, the number of exchanged messages is fix and directly proportional to the search space size, making this property of a problem instance particularly relevant.

References

[BD97] B. Baudot and Y. Deville. Analysis of distributed arc-consistency algorithms. Technical Report RR-97-07, U. Catholique Louvain, 1997.

[BMM01] C. Bessière, A. Maestre, and P. Meseguer. Distributed dynamic backtracking. In *CP*, page 772, 2001.

[BOGW88] M. Ben-Or, S. Goldwasser, and A. Widgerson. Completeness theorems for non-cryptographic fault-tolerant distributed computating. In *STOC*, pages 1–10, 1988.

[CDK91] Z. Collin, R. Dechter, and S. Katz. On the feasibility of distributed constraint satisfaction. In *Proceedings of IJCAI 1991*, pages 318–324, 1991.

[FBKG02] C. Fernàndez, R. Béjar, B. Krishnamachari, and C. Gomes. Communication and computation in dis. CSP algorithms. In *CP*, pages 664–679, 2002.

[FMG02] B. Faltings and S. Macho-Gonzalez. Open constraint satisfaction. In *CP*, 2002.

[FMW01] E.C. Freuder, M. Minca, and R.J. Wallace. Privacy/efficiency tradeoffs in distributed meeting scheduling by constraint-based agents. In *Proc. IJCAI DCR*, pages 63–72, 2001.

[Gol04] Oded Goldreich. *Foundations of Cryptography*, volume 2. Cambridge, 2004.

[GP02] IP. Gent and P. Prosser. An empirical study of the stable marriage problem with ties and incomplete lists. In *ECAI 2002*, pages 141–145, 2002.

[Ham99] Youssef Hamadi. *Traitement des problèmes de satisfaction de contraintes distribués*. PhD thesis, Université Montpellier II, Juillet 1999.

[HCN$^+$01] T Herlea, J. Claessens, G. Neven, F. Piessens, B. Preneel, and B. Decker. On securely scheduling a meeting. In *Proc. of IFIP SEC*, pages 183–198, 2001.

[IM02] R. Irving and D. Manlove. The stable roommates with ties. *Journal of Algorithms*, 43(1):85–105, 2002.

[Kil88] J. Kilian. Founding cryptography on oblivious transfer. In *Proc. of ACM Symposium on Theory of Computing*, pages 20–31, 1988.

[LV97] Michel Lemaître and Gérard Verfaillie. An incomplete method for solving distributed valued constraint satisfaction problems, 1997.

[MJ00] P. Meseguer and M. Jiménez. Distributed forward checking. In *CP'2000 Distributed Constraint Satisfaction Workshop*, 2000.

[MR99] Eric Monfroy and Jean-Hugues Rety. Chaotic iteration for distributed constraint propagation. In *SAC*, pages 19–24, 1999.

[MTSY04] P.J. Modi, M. Tambe, W.-M. Shen, and M. Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *AIJ*, 2004.

[SGM96] G. Solotorevsky, E. Gudes, and A. Meisels. Algorithms for solving distributed constraint satisfaction problems (DCSPs). In *AIPS96*, 1996.

[Sha79] A. Shamir. How to share a secret. *Comm. of the ACM*, 22:612–613, 1979.

[Sil02] Marius-Călin Silaghi. *Asynchronously Solving Distributed Problems with Privacy Requirements*. PhD Thesis 2601, (EPFL), June 27, 2002. http://www.cs.fit.edu/˜msilaghi/teza.

[Sil03a] M.C. Silaghi. Arithmetic circuit for the first solution of distributed CSPs with cryptographic multi-party computations. In *IAT*, Halifax, 2003.

[Sil03b] M.C. Silaghi. Solving a distributed CSP with cryptographic multi-party computations, without revealing constraints and without involving trusted servers. In *IJCAI-DCR*, 2003.

[Sil04a] M.-C. Silaghi. Secure distributed CSP solvers. http://www.cs.fit.edu/ msilaghi/secure/, 2004.

[Sil04b] M.C. Silaghi. Incentive auctions and stable marriages problems solved with n/2-privacy of human preferences. Technical Report CS-2004-11, FIT, 2004.

[SR04] M. Silaghi and V. Rajeshirke. The effect of policies for selecting the solution of a DisCSP on privacy loss. In *AAMAS*, pages 1396–1397, 2004.

[SSHF00a] M.-C. Silaghi, D. Sam-Haroud, and B. Faltings. Asynchronous search with aggregations. In *Proc. of AAAI2000*, pages 917–922, Austin, August 2000.

[SSHF00b] M.-C. Silaghi, D. Sam-Haroud, and B. Faltings. Asynchronous search with private constraints. In *Proc. of AA2000*, pages 177–178, Barcelona, June 2000.

[WF02] R.J. Wallace and E.C. Freuder. Constraint-based multi-agent meeting scheduling: Effects of agent heterogeneity on performance and privacy loss. In *DCR*, pages 176–182, 2002.

[WS04] R. Wallace and M.C. Silaghi. Using privacy loss to guide decisions in distributed CSP search. In *FLAIRS'04*, 2004.

[Yao82] A. Yao. Protocols for secure computations. In *FOCS*, pages 160–164, 1982.

[YDIK98] M. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE TKDE*, 10(5):673–685, 1998.

[YSH02a] M. Yokoo, K. Suzuki, and K. Hirayama. Secure distributed constraint satisfaction: Reaching agreement without revealing private information. In *Proc. of the AAMAS-02 DCR Workshop*, Bologna, July 2002.

[YSH02b] M. Yokoo, K. Suzuki, and K. Hirayama. Secure distributed constraint satisfaction: Reaching agreement without revealing private information. In *CP*, 2002.

[ZM91] Y. Zhang and A. K. Mackworth. Parallel and distributed algorithms for finite constraint satisfaction problems. In *Proc. of Third IEEE Symposium on Parallel and Distributed Processing*, pages 394–397, 1991.

[ZWW02] W. Zhang, G. Wang, and L. Wittenburg. Distributed stochastic search for constraint satisfaction and optimization: Parallelism, phase transitions and performance. In *PAS*, 2002.