BSCA-P: Privacy Preserving Coalition Formation

Bastian Blankenburg and Matthias Klusch

DFKI - German Research Center for Artificial Intelligence
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
{blankenb,klusch}@dfki.de

Abstract. In the setting of cooperation of rational web service agents via coalition formation, we devise an algorithm BSCA-P to form recursively bilateral Shapley value stable coalitions. The main focus lies on privacy aspects: we show that the BSCA-P enables the formation of subgame stable and individually rational coalitions while hiding absolute coalition values and payoffs, as well as allowing for anonymous service requests and access.

1 Introduction

Coalition formation (CF) is the coming together of a number of distinct, autonomous agents in order to act as a coherent grouping in which they increase their individual gains by collaborating. As such, it is an important form of interaction in multi-agent systems (MAS) and has been advocated in task allocation scenarios[9]. In this context, cooperative game theory (CGT) provides a well developed and mathematically founded framework to determine which coalitions should be formed and how the respective coalition values should be distributed in an individually rational and stable manner [5] (i.e. no agents has an incentive to break away from its coalition).

In recent years, a number of CF methods which account for different real-world problems have been proposed in the literature. Examples include [2, 4] for CF under uncertainty of coalition values, or [11, 1] which consider inter-agent trust in CF. The development of privacy preserving CF protocols, however, has not received much attention yet. This is the problem that we address in this paper for the setting of cooperative web service agents. More precisely, we present a CF protocol which allow participating agents to

1. hide its (exact) payoff from all but one other agents,
2. hide its individual utility from all other agents,
3. anonymously request and access offered services,
4. hide the fact that a service from a specific agent has been accessed also from other agents and
5. hide input/output data for services from all agents except the recipient.

The remainder of this paper is organized as follows: in section 2 we introduce our model of web service agents and the coalition game model. In section 3 we show how this model can be exploited to negotiate coalitions while hiding information about coalition values, local worths and payoffs. In section 4 we adopt an anonymous routing protocol to enable anonymous service access and introduce other notions of anonymity. Finally, we propose and discuss the coalition formation protocol BSCA-P in section 5, and conclude in section 6.

2 Coalitions of Web Service Agents

We define a *web service* as any computational process for which all input and output data can be transferred over the internet. If a web service involves the execution of other web services, it is called a *composed web service*. Otherwise, it is called a *primitive web service*. A formal specification of a web service in an appropriate language L is called a *web service description* (L-WSD). Examples for L include WSDL for traditional web services or OWL-S for semantic web services. A message containing an offer to execute a web service ws and an L-WSD of ws is called a *web service advertisement* (L-WSA). A message containing a request for the execution of a web service ws and an L-WSD of ws is called a *web service request* (L-WSR). A comparison of two L-WSDs in order to find out wether an advertised web service and a requested web service match is called a *web service matching* (L-WSM). We then define an *L-web service agent* as an agent which

1. offers any number (including zero) of web services,
2. is able to send L-WSAs for its offered web services,
3. requests any number (including zero) of web services,
4. possesses L-WSDs of its requested web services and
5. is able to perform L-WSMs.

In the following, we consider only sets of web service agents using the same language L, and thus omit the 'L' in our notation. Also, instead of 'web service agent', we also say just 'agent'. We denote by R_a the set of all requests by agent a, and by OS_a the set of all offered services by agent a.

For simplicity, we assume that each agent only offers primitive web services. However, the requesting agents might compute compositions of these primitive services. We further assume each agent a to have a certain private monetary valuation $w_a(WS)$ for the accomplishment of each service it requests. Finally, the execution of service WS by an agent a has a cost $c_a(WS)$.

We can now model this setting as a *coalition game*. Let $\mathcal{A}$ denote the set of all agents in a given system. We call subsets $C \subseteq \mathcal{A}$ executing services for each other a *coalition*. Let $E_a(C)$ denote the set of all services executed by a, and $R_a(C)$ the set of all services of other members of C which are accessed by a. Then, a's immediate monetary result (that is, without side-payments) of being a member of C, which we call *local worth* of a in C, is determined by

$$lw_a(C) := \sum_{WS \in R_a(C)} w_a(WS) - \sum_{WS \in E_a(C)} c_a(WS) \tag{1}$$

Thus, we can define an overall value

$$v(C) := \sum_{a \in C} lw_a(C) \tag{2}$$

of C, which we call C's *coalition value*. The pair $(\mathcal{A}, v)$ then defines the coaltion game. In cooperative game theory, coalitions may not overlap. A *configuration* $(\mathcal{S}, u)$ for a game $(\mathcal{A}, v)$ specifies a *payoff distribution* $u : A \mapsto \mathbb{R}$ for a *coalition structure* $\mathcal{S}$, a partition of A. $u(a), a \in A$ denotes the *payoff* for agent a. u is called *individually rational* iff $\forall a \in A : u(a) \geq v(a)$ and *efficient* iff $\forall C \in \mathcal{S} : \sum_{a \in C} u(a) = v(C)$ We also write $\forall C \subseteq \mathcal{A} : u(C) := \sum_{a_i \in C} u_i$.

In order to implement a payoff distribution u, each agent generally will have to make/receive side-payments. Keeping in mind the local worths we define the total amount of side-payment that a has to receive from other agents in C as

$$sp_u(a, C) := u(a) - lw_a(C) \tag{3}$$

Of course, $sp_u(a, C)$ can be negative, meaning that a has to make a side-payment of $|sp_u(a, C)|$ to other agents in C. We also write for $C^* \subseteq C$:

$$sp_u(C^*, C) := \sum_{a \in C^*} sp_u(a, C) \tag{4}$$

If $C^* = C$, we just write $sp_u(C)$

Corollary 1 *Let $C \in \mathcal{S}$. Then $sp_u(C) = 0$ if an only if u is efficient wrt. $\mathcal{S}$.*

Example 1. Consider a game of three agents: $\mathcal{A} = \{a_1, a_2, a_3\}$. They offer and request services according to table 1. Considerin coalition $C_1 = \{a_1, a_2\}$, we have

agent a	offers	$c_a(.)$	requests	$w_a(.)$
a_1	ws_1	1	ws_2	2
	ws_3	2	ws_3	3
a_2	ws_2	1	ws_4	2
a_3	ws_4	1	ws_1	3

Table 1. Offered/requested services in example game

$lw_{a_1}(C_1) = w_{a_1}(ws_2) + w_{a_1}(ws_3) - c_{a_1}(ws_3) = 3$, $lw_{a_2}(C_1) = -c_{a_2}(ws_2) = -1$ and $v(C_1) = 2$.

A solution to a game is given by an individually rational and efficient configuration which satisfies a chosen *stability concept*. Unfortunately, the classical stability concepts are of high computational complexity, i.e. at least exponential. However, we consider only the case where coalitions are built up by a bilateral merging process. We thus utilize a simplified version of the Shapley value[8], the *(recursive) bilateral* Shapley value:

The union C of two disjunkt coalitions $C_1, C_2 \subset \mathcal{A} \setminus \emptyset$ is called a *bilateral coalition.* C_1 and C_2 are called subcoalitions of C. A bilateral coalition C is called *recursively bilateral* iff it is the root node of a binary tree denoted T_C for which (a) every non-leaf node is a bilateral coalition and its subcoalitions are its children and (b) every leaf node is a single-agent coalition.

A coalition structure $\mathcal{S}$ for $(\mathcal{A}, v)$ is called *(recursively) bilateral* if $\forall C \in \mathcal{S}$: C is (recursively) bilateral or $C = a$, $a \in \mathcal{A}$. The *bilateral Shapley value* $\sigma_b(C, C_i, v)$,$C_i, i \in \{1, 2\}$ in the bilateral coalition C is defined as the Shapley value of C_i in the game $(\{C_1, C_2\}, v)$:

$$\sigma_b(C_i, C, v) = \frac{1}{2}v(C_i) + \frac{1}{2}(v(C) - v(C_k)) \tag{5}$$

with $k \in \{1, 2\}, k \neq i$.

Given a recursively bilateral coalition structure $\mathcal{S}$ for a game $(\mathcal{A}, v)$, a payoff distribution u is called *recursively bilateral Shapley value stable* iff for each $C \in \mathcal{S}$, every non-leaf node C^* in $T_C : u(C_i^*) = \sigma_b(C_i^*, C^*, v_{C^*}), i \in 1, 2$ with $\forall C^{**} \subseteq \mathcal{A}$:

$$v_{C^*}(C^{**}) = \begin{cases} \sigma_b(C_k^p, C^p, v_{C^p}) \text{ if } C^p \in T_C, C^* = C^{**} = C_k^p, \\ \qquad\qquad\qquad\qquad k \in 1, 2 \\ v(C^{**}) \qquad\qquad \text{otherwise} \end{cases} \tag{6}$$

In other words, for a merge of two recursively bilateral coalitions, the coalition value is distributed down the coalition tree applying the bilateral Shapley value to the actual payoffs of the respective parent coalitions instead of their coalition values.

Example 2. Consider again the game from example 1 and the bilateral coalition $C_1 = \{a_1\} \cup \{a_2\}$. Since $v(\{a_1\}) = 1$ and $v(\{a_2\}) = 0$, we have $\sigma_b(\{a_1\}, \{a_1\} \cup \{a_2\}, v) = \frac{1}{2} + \frac{1}{2}(2 - 0) = 1.5$ and $\sigma_b(\{a_2\}, \{a_1\} \cup \{a_2\}, v) = \frac{1}{2}(2 - 1) = 0.5$

Now consider a merge of C_1 with $C_2 = \{a_3\}$ ($C = C_1 \cup C_2$). We have $v(C) = 5$ and $v(C_2) = 0$, thus $\sigma_b(C_1, C, v) = 1 + \frac{1}{2}5 = 3.5$ and $\sigma_b(C_2, C, v) = \frac{1}{2}(5 - 2) = 1.5$

For a recursively bilateral Shapley value stable payoff distribution we have to consider v^* with $v^*(\{a_1, a_2\}) = 3.5$ and for all other coalitions $v^*(C) = v(C)$: $u(a_1) = \sigma_b(\{a_1\}, \{a_1\} \cup \{a_2\}, v^*) = \frac{1}{2} + \frac{1}{2}(3.5 - 0) = 2.25$ and $u(a_2) = \sigma_b(\{a_2\}, \{a_1\} \cup \{a_2\}, v^*) = \frac{1}{2}(3.5 - 1) = 1.25$.

3 Hiding Local Worths and Coalition Values

In this section we show that the recursively bilateral Shapley value is well-suited when hiding coalition values and local worths. It is easy to see that (5) can be rewritten as

$$\sigma_b(C_i, C, v) = v(C_i) + \frac{1}{2} \cdot (v(C) - v(C_1) - v(C_2)) \tag{7}$$

with $i \in \{1, 2\}$. Thus, the *additional value*

$$av(C_1, C_2) := v(C_1 \cup C_2) - v(C_1) - v(C_2) \tag{8}$$

produced by forming coalition $C_1 \cup C_2$ is evenly distributed among C_1 and C_2. For recursively bilateral Shapley value stable payoff distributions, this means that each child node in the coalition tree gets half of the additional payoff of its parent node. The share of the total payoff that a node gets is thus directly dependent on its depth in the tree, which is shown by the following lemma.

Lemma 1 *Let $(\mathcal{S}_1, u_1)$ and $(\mathcal{S}_2, u_2)$ configurations for a game $(\mathcal{A}, v)$, with u_1 and u_2 being recursively bilateral Shapley value stable, and $\exists C_1, C_2 \in \mathcal{S}_1 : C = C_1 \cup C_2 \in \mathcal{S}_2$. Then*

$$\forall C^* \in T_C : u_2(C^*) = u_1(C^*) + \frac{av(C_1, C_2)}{2^{d(C^*, T_C)}}$$

Proof. We use induction over $d(C^*, T_C)$: the case $d(C^*, T_C) = 0$ is obvious because of the efficiency of σ_b and the definition of av.

For $d(C^*, T_C) = 1$, we have $C^* = C_i$, $i \in \{1, 2\}$ and $u_2(C_i) = \sigma_b(C_i, C, v) = v(C_i) + \frac{1}{2} av(C)$. Again because of the efficiency of σ_b, $v(C_i) = u_1(C_i)$, and thus $v(C_i) + \frac{1}{2} av(C) = u_1(C_i) + \frac{av(C)}{2^{d(C^*, T_C)}}$.

For $d(C^*, T_C) = k > 1$, assuming the lemma is true for all C^{**} with $d(C^{**}, T_C) < k$, we have $C^* = C_i^p$, $i \in \{1, 2\}$, $C^p \in T_C$, $d(C_i^p, T_C) = d(C^p, T_C) + 1$ and $u_2(C_i^p) = \sigma_b(C_i^p, C^p, v_{C_i})$ with $v_{C_i^p}(C^p) = u_2(C^p) = u_1(C^p) + \frac{av(C)}{2^{d(C^p, T_C)}}$. Applying 6 and 7, we get

$$\begin{aligned}
u_2(C_i^p) &= v(C_i^p) + \frac{1}{2}(u_2(C^p) - v(C_i^p) - v(C_k^p)) \\
&= v(C_i^p) + \frac{1}{2}(u_1(C^p) + \frac{av(C)}{2^{d(C^p, T_C)}} - v(C_i^p) - v(C_k^p)) \\
&= v(C_i^p) + \frac{1}{2}(u_1(C^p) - v(C_i^p) - v(C_k^p)) + \frac{av(C)}{2^{d(C^p, T_C)+1}} \\
&= u_1(C_i^p) + \frac{av(C)}{2^{d(C_i^p, T_C)}}
\end{aligned}$$

For the merge of C_1 and C_2 to form $C = C_1 \cup C_2$, we further definine the *additional local worth* of agent $a \in C_i$, $i \in \{1, 2\}$:

$$alw_a(C_i, C) := lw_a(C) - lw_a(C_i), \tag{9}$$

and the summarized additional local worth for a subcoalition $C^* \in T_{C_i}$

$$alw(C^*, C_i, C) := \sum_{a \in C^*} (alw_a(C_i, C)) \tag{10}$$

Also, note that

$$\begin{aligned}
av(C_1, C_2) &= \sum_{a \in C} lw_a(C) - \sum_{a \in C_1} lw_a(C_1) - \sum_{a \in C_2} lw_a(C_2) \\
&= alw(C_1, C_1, C) + alw(C_2, C_2, C)
\end{aligned} \tag{11}$$

The following theorem shows that in order to compute its side-payment when merging coalitions C_1 and C_2, each subcoalition $C^* \in T_{C_i}$ only needs to consider its side-payment for the case without the merge, the additional value $av(C_1, C_2)$ and its additional local worth $alw(C^*, C_i, C)$:

Theorem 1 *Let $(\mathcal{S}_1, u_1)$ and $(\mathcal{S}_2, u_2)$ configurations for a game $(\mathcal{A}, v)$, with u_1 and u_2 being recursively bilateral Shapley value stable, and $\exists C_1, C_2 \in \mathcal{S}_1 : C = C_1 \cup C_2 \in \mathcal{S}_2$. Then $\forall C^* \in T_{C_i}$, $i \in \{1, 2\}$:*

$$sp_{u_2}(C^*, C) = sp_{u_1}(C^*, C_i) + \frac{alw(C_1, C_1, C) + alw(C_2, C_2, C)}{2^{d(C^*, T_C)}} - alw(C^*, C_i, C)$$

Proof. Remember that for any u, $sp_u(C^*, C) = \sum_{a \in C^*} u(a) - lw_a(C) = u(C^*) - \sum_{a \in C^*} lw_a(C)$ (see 4). Because of lemma 1, 9 , 10 and 11, we can rewrite

$$\begin{aligned}
sp_{u_2}(C^*, C) &= u_1(C^*) + \frac{av(C_1, C_2)}{2^{d(C^*, T_C)}} - \sum_{a \in C^*} lw_a(C) \\
&= u_1(C^*) + \frac{av(C_1, C_2)}{2^{d(C^*, T_C)}} - \sum_{a \in C^*} (lw_a(C_i) + alw_a(C_i, C)) \\
&= sp_{u_1}(C^*, C_i) + \frac{av(C_1, C_2)}{2^{d(C^*, T_C)}} - alw(C^*, C_i, C) \\
&= sp_{u_1}(C^*, C_i) + \frac{alw(C_1, C_1, C) + alw(C_2, C_2, C)}{2^{d(C^*, T_C)}} - alw(C^*, C_i, C)
\end{aligned}$$

Please note that in the case of $C^* = C_i$, $sp_{u_1}(C^*, C_i) = 0$ because $C_i \in \mathcal{S}_1$ and corollary 1. It is thus clear that in order to obtain recursively bilateral Shapley value stable payoff distributions by repeatedly merging coalitions, subcoalitions have to inform each other only about their additional local worths. Absolute local worths need not to be communicated, and absolute coalition values do not have to be known at all.

The results of this section are employed in the specification of the coalition formation protocol BSCA-P in section 5, but we give an example here:

Example 3. Consider again the situation from example 2. At first $\{a_1\}$ and $\{a_2\}$ merge to form C_1, with $alw_{a_1}(\{a_1\}, C_1) = 3 - 1 = 2$ and $alw_{a_2}(\{a_2\}, C_1) = -1 - 0 = -1$. According to theorem 1 we get

$$sp_u(\{a_1\}) = 0 + \frac{2 + (-1)}{2^1} - 2 = -1.5 \text{ and}$$
$$sp_u(\{a_2\}) = 0 + \frac{2 + (-1)}{2^1} - (-1) = 1.5$$

Thus, the net amount received by a_1 is

$$u(a_1) = lw_{a_1}(C_1) + sp_u(\{a_1\}) = 3 - 1.5 = 1.5 = \sigma_b(\{a_1\}, \{a_1\} \cup \{a_2\}, v)$$

and that of a_2 is

$$u(a_2) = lw_{a_2}(C_1) + sp_u(\{a_2\}) = -1 + 1.5 = 0.5 = \sigma_b(\{a_2\}, \{a_1\} \cup \{a_2\}, v)$$

Second, C_1 merges with $C_2 = \{a_3\}$ to form $C = C_1 \cup C_2$. By looking at the service offers and requests, the agents (and coalition C_1 determine their additional local worths:

$$
\begin{aligned}
&alw_{a_1}(\{a_1\}, C) = 2 - 3 = -1,\\
&alw_{a_2}(\{a_2\}, C) = 1 + 1 = 2,\\
&alw(C_1, C_1, C) = alw_{a_1}(\{a_1\}, C) + alw_{a_2}(\{a_2\}, C) = 1 \text{ and}\\
&alw(C_2, C_2, C) = 2 - 0 = 2
\end{aligned}
$$

The additional coalition value is thus

$$av(C_1, C_2) = alw(C_1, C_1, C) + alw(C_2, C_2, C) = 3$$

Applying theorem 1 again, we get for the new payoff distribution u^*

$$
\begin{aligned}
sp_{u^*}(C_1) &= 0 + \frac{1+2}{2^1} - 1 = 0.5 \text{ and}\\
sp_{u^*}(C_2) &= 0 + \frac{1+2}{2^1} - 2 = -0.5.
\end{aligned}
$$

The net payoffs of C_1 and C_2 are of course equal to their resp. bilateral Shapley values:

$$
\begin{aligned}
u^*(C_1) &= lw_{a_1}(C) + lw_{a_2}(C) + sp_{u^*}(C_1)\\
&= 2 + 1 + 0.5 = 3.5 = \sigma_b(C_1, C, v) \text{ and}\\
u^*(C_2) &= lw_{a_3}(C) + sp_{u^*}(C_2) = 2 - 0.5 = 1.5 = \sigma_b(C_2, C, v)
\end{aligned}
$$

For the side-payments within C_1 we again apply theorem 1:

$$
\begin{aligned}
&sp_{u^*}(\{a_1\}, C) = sp_u(\{a_1\}, C_1) + \frac{1+2}{2^2} - (-1) = -1.5 + 0.75 + 1 = 0.25 \text{ and}\\
&sp_{u^*}(\{a_2\}, C) = 1.5 + 0.75 - 2 = 0.25
\end{aligned}
$$

4 Anonymous Service Access

In this section, we introduce some anonymity and encryption concepts that enable anonymous and secure web service access.

To achieve this, we use an anonymous communication protocol based on rerouting. In a rerouting protocol, a message is not directly sent to the receiver, but travels over intermediate network nodes, or agents in our case. The specific protocol we utilize is roughly based on onion routing [10]. It was originally defined for HTTP-connections, but we adapt it here for our agent coalition formation setting, by looking only at high-level messages sent between the agents instead of technical details of an underlying protocol. Our focus is to enable the agents to request and access services within their coalition anonymously. We thus also do

not bother about problems like possible eavesdropper agents or traffic analysis, as such problems are out of scope of this paper.

The basic idea of the onion routing protocol is to wrap a message in several layers of encryption and reroute it over several rerouting nodes such that no single node is able to determine the sender and receiver of a message. Also, when one agent contacts another, the nodes over which messages are sent are chosen randomly. Figure 1 illustrates this for a three-agent case. It incorpo-

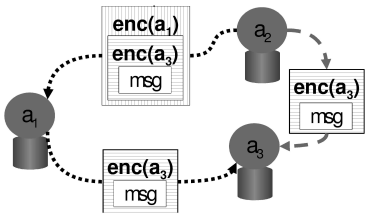

Fig. 1. Two ways of a_2 contacting a_3 via Onion Routing.

rates a public/private key encryption method, such as the well-known RSA method (originally proposed in [7]). Thus, we extend our agent model such that every agent a is required to possess a private key $privkey_a$ and a matching public key $pubkey_a$ for the chosen encryption method. Further, a needs to be able to execute according encryption/decryption functions. In the following, $enc(pubkey, m)$ denotes a function that encrypts message m using the public key $pubkey$, and $dec(privkey, em)$ denotes the corresponding decryption function for the encrypted message em using the private key $privkey$. To let agent a_1 send an enctypted message m to agent a_2, a_1 encrypts m by executing $enc(pubkey_{a_2}, m)$, sends the result em to a_2 which decypts it by executing $dec(privkey_{a_2}, em$. Thus, the agents need to perform an initial public key exchange. In the onion protocol, actually only a part of a message is encrypted with the public key method. This part contains a key for a symmetric encryption method, i.e. one that uses the same key for encryption and decryption. The remainder of the message is encrypted with this method. This is done because of performance reasons, since symmetric encryption methods usually are much faster than public key methods. However, we go not into those details here, and consider such optimizations as part of the implementation of the *enc* and *dec* functions.

We are now ready to define our anonymous message sending algorithm:

Algorithm 1 *To anonymously send a message m to agent a_2 over i intermediate agents, agent a_1 performs the following:*

1. *Randomly generate an ordered list L with length $i+1$ of agents, such that $L_j \neq L_{j+1}$, $\forall 1 \leq j < i$, and $L_{i+1} = a_2$, where L_j is the agent at position j in L.*
2. *Set $em_{i+1} := enc(pubkey_{a_2}, m)$.*

3. *For $l = i$ to 1 do:*
   (a) *Set $m^* := (L_{l+1}, em_{l+1})$.*
   (b) *Set $em_l := enc(pubkey_{L_l}, m^*)$.*
4. *Send em_1 to L_1.*

For this to work, each agent also needs to implement an algorithm to handle incoming encrypted messages:

Algorithm 2 *When receiving an encrypted message em, agent a_1 performs:*

1. *Set $m := dec(privkey_{a_1}, em)$*
2. *If m is of the form (a_2, em), $a_2 \in \mathcal{A}$, send em to a_2; else process m like an incoming unencrypted message.*

To measure a degree of anonymity, different notions have been proposed in the literature, such as total or group anonymity, under possibilistic or probabilistic interpretations (see e.g. [6, 3]. Here, we will apply the concept of possibilistic *agent k-anonymity*, which requires only that there exists some set of agents K with size k, such that each $a \in K$ is a possible sender. Specifically, this anonymity is measured in the following way. When the two coalitions C_1 and C_2 perform a merge to form $C = C_1 \cup C_2$, they need to compute and inform each other about $alw(C_1, C_1, C)$ and $alw(C_2, C_2, C)$ (see section 3). Because of the definition of the local worths, $alw(C_i, C_i, C) > 0$, $i \in \{1, 2\}$, means that in coalition C_i, more worth is produced by agents getting services executed than costs are produced due to agents executing services. All Agents in C thus can infer that at least one agent $a \in C_i$ accesses a service in C_k, $k \in \{1, 2\}, k \neq i$. We thus obtain the degree of agent k-anonymity for agents in C_i wrt. agents in C_k :

$$aa(C_i, C_k) = |C_i|$$

However, the degree of agent k-anonymity of a wrt. other agents in C_i is in general only $k = 1$. This is because in order to compute $alw(C_i, C_i, C)$, each subcoalition $C^* \in T_{C_i}$ has to compute $alw(C^*, C_i, C)$ first. In particular, agent a has in general to inform some other agent in C_i about $alw(\{a\}, C_i, C)$.

Thus, we also use the concept of *service k-anonymity*, expressing that an agent a accesses any one of k possible services. In the case of agent $a \in C_i$ accessing a service in C_k, the degree of service k-anonymity for a wrt. to the agents in C_i is equal to the total number of services offered by agents in C_k:

$$sa(C_i) = |\bigcup_{a \in C_k} OS_a|$$

In the following, we assume that each agent maintains minimum k-anonymity degrees $aa_{min}(WS) \in \mathbb{N}$ and $sa_{min}(WS) \in \mathbb{N}$ for each service it is interested in requesting. When forming the coalition C, agent $a \in C_i$ then only requests a service WS from an agent in C_k if these minimum degrees are met, i.e. $WS \in R_a(C)$ if

$$aa(C_i, C_k) \geq aa_{min}(WS) \text{ and} \tag{12}$$

$$sa(C_i) \geq sa_{min}(WS) \tag{13}$$

hold.

5 Coalition Formation Protocol BSCA-P

In this section, we finally propose the coalition formation protocol BSCA-P applying the concepts that have benn introduced in the previous sections. In the BSCA-P, each coalition is represented by one agent which is responsible for the communication with other coalitions. To simplify the choice of a representative, we assume there exists an ordering function o defined on the set of all agents. We also assume that service offers, along with the service execution costs, are made public beforehand (e.g., by broadcasting).

Algorithm 3 *For a game* $(\mathcal{A}, v)$, $\mathcal{S}_0 := \{\{a\} | a \in \mathcal{A}\}$, $r := 0$ *and* $\forall C \in \mathcal{S}_0 : sp_0(C) := 0$. *In every coalition* $C \in \mathcal{S}_r$, *every agent* $a \in C$ *performs:*

1. *Let* $C \in \mathcal{S}_r$, $a \in C$ *and* $\mathcal{S}^* := \mathcal{S} \setminus C$.
2. Communication:
   (a) *For all* $C^* \in \mathcal{S}^*$ *do:*
      i. *Determine* $R_a(C^*)$ *using the sets* OS_{a^*} *for each* $a^* \in C^*$, *accounting for costs and ensuring compliance with 12 and 13.*
      ii. *For each service request which is both in* $R_a(C)$ *and* $R_a(C^*)$, *keep only the least costly one.*
      iii. *Set* $alws_a(C^*) := alw_a(C, C^*)$.
      iv. *For each bilateral coalition* C^a, $C^a \in T_C$, $a \in C^a$, $a = Rep(C_a)$, *wait for a message from* $Rep(C_i^a)$, $i \in 1, 2, a \notin C_i^a$ *containing* $alws_{Rep(C)}(C^*)$ *and set* $alws_a(C^*) := alws_a(C^*) + alws_{Rep(C)}(C^*)$.
      v. *If* $a = Rep(C)$ *then send* $alws_a(C^*)$ *to* $Rep(C^*)$; *else send* $alws_a(C^*)$ *to* $Rep(C^+)$ *with* $C^+ \in T_C$, $a = Rep(C_i^+)$, $i \in 1, 2, a \neq Rep(C^+)$.
   (b) *If* $a = Rep(C)$ *then receive* $alws_{Rep(C^*)}(C)$ *and set* $alws(C^*) := alws_{Rep(C^*)}(C) + alws_a(C^*)$ *for all* $C^* \in \mathcal{S}^*$; *else go to step 3i.*
3. Coalition Proposals:
   (a) *Set* $Candidates := \mathcal{S}^*$, $New := \emptyset$ *and* $Obs := \emptyset$
   (b) *Determine a coalition* $C^+ \in Candidates$ *with* $\forall C^* \in Candidates : alws_a(C^+) \geq alws_a(C^*)$.
   (c) *Send a proposal to* $Rep(C^+)$ *to form coalition* $C \cup C^+$.
   (d) *Receive all coalition proposals from other agents.*
   (e) *If no propsal from* $Rep(C^+)$ *is received and* $Candidates \neq \emptyset$, *set* $Candidates := Candidates \setminus \{C^+\}$ *and go to step 3b.*
   (f) *If a proposal from* $Rep(C^+)$ *is received, then form the coalition* $C \cup C^+$:
      i. *If* $o(Rep(C)) < o(Rep(C^+))$ *then set* $Rep(C \cup C^+) := Rep(C)$; *else set* $Rep(C \cup C^+) := Rep(C^+)$.
      ii. *Inform all other* $Rep(C^*), C^* \in \mathcal{S}^* \setminus C^+$ *and all* $a^* \in C$, $a^* \neq a$ *about the new coalition and* $Rep(C \cup C^+)$
      iii. $New := \{C \cup C^+\}$, $Obs := \{C, C^+\}$
   (g) *Receive all messages about new coalitions. For each new coalition* $C_1 \cup C_2$ *and* $Rep_{C_1 \cup C_2}$, *set* $Candidates := Candidates \setminus \{C_1, C_2\}$, $New := New \cup \{C_1 \cup C_2\}$ *and* $Obs := Obs \cup \{C_1, C_2\}$.
   (h) *Send the sets* New *and* Obs *to all other coalition members* $a^* \in C$, $a^* \neq a$

(i) If $a \neq Rep(C)$ then receive the sets New and Obs from $Rep(C)$.
(j) Set $r := r + 1$, $\mathcal{S}_r := (\mathcal{S}_{r-1} \setminus Obs) \cup New$.
(k) For each (sub-)coalition $C^* \in T_C$ with $Rep(C^*) = a$, determine $sp_r(C^*)$ according to theorem 1 (using $sp_{r-1}(C^*)$ instead of $sp_u(C^*)$).
(l) If $C_r = C_{r-1}$ then stop; else go to step 2

Theorem 2 *With $n = |\mathcal{A}|$ and $m := max_{a \in \mathcal{A}}\{|R_a|\}$, the computational complexity of the BSCA-P is in $O(n^3m^2)$.*

Proof. In any round r, $\mathcal{S}_r \leq n$. The iteration in step 2a is thus done at most n times. In step 2(a)i, for each service in R_a, a has to find an agent in the potential partner coalition which offers this service at the least cost. The conditions 12 and 13 only have to be checked once for each service, for which we assume negligible complexity. Thus, at most nm operations are required in this step. Step 2(a)ii can be done in less than m^2 steps. All other steps within and outside of the iteration in step 2a are of less complexity. Thus, the complexity of one round of the BSCA-P is in $O(n)(O(nm) + O(m^2)) = O(n^2m^2)$. Since the maximum number of coalition merges is smaller than n (because after at most $n-1$ merges, the grand coalition is formed), the number of rounds is also bound by n. The overall complexity of the BSCA-P is thus $O(n)O(n^2m^2) = O(n^3m^2)$.

Theorem 3 *In the BSCA-P, the number of messages sent by an agent is in $O(n^2)$.*

Proof. During each iteration in step 2(a)i, in step 2(a)v a message to the agent's subcoalition representative or to $Rep(C^*)$. Assuming that agents which are representatives of several subcoalitions omit sending messages to themselves, and with at most n iterations in step 2(a)i (see above), the number of messages sent during the iteration is in $O(n)$. The number of messages sent in step 3(f)ii is also in $O(n)$. Thus, with at most n rounds, the overall number of messages sent by an agent in the BSCA-P is in $O(n^2)$

When the protocol is finished and thus coalitions are formed, agents still have to execute the following steps in order to implement the coalitions:

1. Each agent runs algorithm 2 continously in order to enable anonymous service access.
2. Concurrently, algorithm 1 is executed by the agents requesting services to actually access these services at their providers.
3. All (sub-)coalition representatives execute their respective side-payments sp_r for their (sub-)coalitions. Each representative only makes/receives payments to/from representatives of immediate parent and child coalitions, such that no additional information about payments is gained by any agent.

The last step ensures that only a representative of a two-agent coalition is informed about individual side-payments, and only about two of them: its own, and the other agent of the two-agent coalition. Therefore, only the first partner agent that an agent a coalesces with might ever know a's exact side-payment. However, the individual utilities still remain hidden from all other agents.

6 Conclusions

In this paper, a privacy-preserving coalition formation protocol was proposed. We have shown that in order to form recursively bilateral Shapley value stable coalitions, individual payoffs may be hidden from most agents while individual utilites can be completely hidden, and absolute coalition values need not to be known at all. Also, we showed that within a coalition, services can be accessed by cooperating agents with certain degrees of anonymity. Thus, agents can hide the fact that they access specific services even from agents which are members of the same coalition.

References

1. B. Blankenburg, R.K. Dash, S.D. Ramchurn, M. Klusch, and N.R. Jennings. Trusted kernel-based coalition formation. In *Proc. 4th Int. Conf. on Autonomous Agents and Multi-Agent Systems*, Utrecht, Holland, 2005. to appear.
2. Georgios Chalkiadakis and Craig Boutilier. Bayesian reinforcement learning for coalition formation under uncertainty. In *Proc. 3rd Int. Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), New York, USA*, New York, USA, 2004. ACM Press.
3. Joseph Halpern and Kevin O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*, Special Edition on CSFW 16:75–88, 2003.
4. S. Kraus, O. Shehory, and Gilad Taase. The advantages of compromising in coalition formation with incomplete information. In *Proc. 3rd Int. Conference on Autonomous Agents and Multiagent Systems (AAMAS 2004), New York, USA*, New York, USA, 2004. ACM Press.
5. Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory*. MIT Press, Cambridge MA, USA, 1994.
6. A. Pfitzmann and M. Köhntopp. Anonymity, unobservability and pseudonymity: a proposal for terminology. In *International Workshop on Designing Privacy Enhancing Technologies*, pages 1–9, New York, 2001. Springer-Verlag.
7. Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 26(1):96–99, 1983.
8. L. S. Shapley. A value for n-person games. In H. W. Kuhn and A. W. Tucker, editors, *Contributions to the Theory of Games II*, volume 28 of *Annals of Mathematics Studies*, pages 307–317. Princeton University Press, Princeton, 1953.
9. O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence Journal*, 101 (1-2):165–200, May 1998.
10. P F Syverson, D M Goldschlag, and M G Reed. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy*, pages 44–54, Oakland, California, 4–7 1997.
11. J. Vassileva, S. Breban, and M. Horsch. Agent reasoning mechanism for long-term coalitions based on decision making and trust. *Computational Intelligence*, 4(18):583–595, 2002.