# Or Best Offer: A Privacy Policy Negotiation Protocol*

Dan D. Walker, Eric G. Mercer, and Kent E. Seamons
Brigham Young University, Provo, UT
{ danl4, egm, seamons } @cs.byu.edu

## Abstract

*Privacy policy languages, such as P3P, allow websites to publish their privacy practices and policies in machine readable form. Currently, software agents designed to protect users' privacy follow a "take it or leave it" approach that is inflexible and gives the server ultimate control. Privacy policy negotiation is one approach to leveling the playing field by allowing a client to negotiate with a server to determine how that server collects and uses the client's data. We present a privacy policy negotiation protocol, "Or Best Offer", that includes a formal model for specifying privacy preferences and reasoning about privacy policies. The protocol is guaranteed to terminate within three rounds of negotiation while producing policies that are Pareto-optimal, and thus fair to both the client and the server.*

## 1 Introduction

Reports of identity theft and the loss and misuse of personal information fuel increased privacy concerns for users. To help alleviate these concerns, many websites publicize their privacy practices. Some use the Platform for Privacy Preferences (P3P) [7], an XML language designed to specify how sites intend to handle information they collect about their visitors. Usually, a site publishes its P3P policy in a well-known location so that client software acting on the user's behalf examines the policy and compares it to the preferences the user has configured to express how her data is to be used. If the policy meets the client's preferences, the software agent approves the transaction, and the user continues browsing without any noticeable interruption.

This "take it or leave it" approach is too limited. Spiekermann et al. [5] have shown that users have a variety of goals in mind when formulating privacy preferences and that almost all are willing to make concessions. Users seem to have an ideal set of preferences that they adhere to when possible, and another set of "good enough" preferences that they are willing to accept for minimal privacy protection. From the server's perspective, although a site may prefer to collect certain types of information and use it in rather promiscuous ways, it may be willing to collect less information, and use it in more protected ways, but only if a user specifically requests such protections.

One way to increase the flexibility of P3P is per-session privacy negotiations [1, 3, 6] that generate fine-grained privacy contracts to govern the use of data collected during a transaction.

Our contributions include the following: a privacy policy negotiation protocol that terminates within a finite number of rounds, a set of preference models that allow for the specification of privacy preferences in a graphical and fairly intuitive fashion, and the application of game theory to specify reasonable utility functions that allow us to show that the protocol is Pareto-optimal, and thus fair to both parties.

The protocol is based on an "Or Best Offer" (OBO) negotiation style, similar to sellers who advertise an item for a fixed price and then express a willingness to entertain a "best offer". The protocol enables per-session privacy contract negotiations that are guaranteed to terminate within a maximum of three negotiation rounds. The server makes a proposal, the client makes a counter-proposal and also gives hints about how the server can best satisfy her needs. Finally, the server does its best to conform to the clients preferences, while at the same time meeting its own needs.

## 2 Related Work

The idea of negotiating per-session privacy policies was rejected in P3P when the designers were unable to envision scenarios in which this capability would be useful. Since then, there have been two proposals.

First, Bennicke, Maaser, and Langendorfer introduced a process by which a privacy contract is proposed and then incrementally modified to meet the demands of both parties. Negotiators make mandatory or optional demands, and the goal is to have all mandatory demands met and as many optional demands as possible. Each party alter-

*This research was supported by funding from the National Science Foundation under grant no. CCR-0325951, prime cooperative agreement no. IIS-0331707, and The Regents of the University of California.

natively assumes the roles, *proposal maker* and *acceptor* [1, 3]. Second, the Privacy Server Protocol Project (PSP) [6] is designed to allow clients and servers to produce "mutual" privacy contracts. These contracts are mutual in the sense that they are considered binding on both the server and the client, instead of applying just to the server, as is usually the case.

There are two limitations present in this earlier work that our research seeks to address. First, earlier specifications allow negotiating parties to engage in potentially endless exchanges of proposals and counter-proposals. Second, negotiating agents in these systems cannot determine whether the concessions they make increase or decrease the chances of a successful negotiation. The goal of this research is a protocol design specification that overcomes these limitations while remaining secure and fair to both parties.

## 3 OBO Protocol Specification

The goals of OBO are to be complete, fair, and secure. A protocol is complete if it always terminates. OBO is complete by definition and only admits three rounds of negotiation. A protocol is fair if it does not favor one party over the other. OBO is fair, and we have proven its fairness using Pareto-optimality from game theory [8]. Pareto-optimality is the notion that in a successful negotiation, neither the client nor the server can better meet their own needs without causing the policy to be worse for the other. A protocol is secure when it protects the negotiating parties or a third party from manipulating the negotiation process. An analysis of OBO security is contained in Section 7.2.

An OBO negotiation consists of three rounds. During each round, one party in the negotiation makes a proposal and the other makes a decision to accept or reject that proposal. The first proposal is issued by the server, in the form of its default privacy policy. This policy details all of the ways in which the server needs to use the client's data to fulfill its purpose, along with other uses that the server might put the data to in order to potentially increase revenue or provide a more customized experience to the user. The client may accept this policy, or issue a counter-proposal. This counter-proposal will remove portions of the policy that the client finds unacceptable. With the counter-proposal, the client is effectively telling the server: "Give me this much privacy, or make me your best offer." Because privacy means different things to different individuals, the client must help the server understand which offers it might formulate are "better" for this particular client. To accomplish this, the client sends information about its preferences to the server, along with its counter-proposal. The server can either accept the client's counter-offer, or it may use the information about the client's preferences to formulate the final "best offer" proposal. If the client rejects the final proposal, then the negotiation fails. If at any point during the negotiation one of the parties accepts a proposal, then the negotiation succeeds and terminates.

$$
\begin{aligned}
Proposal_j = Proposal\{&Pol_j, t\_stamp, ID_c, ID_s \\
&[, Preferences], \\
&sign(Pol_j, t\_stamp, ID_c, ID_s[, Preferences])\} \\
Accept_j = Accept\{&t\_stamp, sign(Proposal_j, t\_stamp)\} \\
Reject = Reject\{&\}
\end{aligned}
$$

**Figure 1. OBO negotiation messages.**

Negotiations are carried out piecewise over portions of the policy referred to as *terms*. An OBO negotiation is a set of simultaneous term-level negotiations carried out in parallel. If all term-level negotiations succeed, the results are combined to produce a complete privacy policy contract. If any of the negotiations fail, the overall negotiation fails.

There are three types of messages in the OBO protocol (see Figure 1).

$Proposal$ The proposal message for round $j$ ($Proposal_j$) contains a proposal policy ($Pol_j$), which consists of all of the terms still under negotiation, as well as any that have been accepted. The message also contains a timestamp representing the date and time the message was created, and tokens that uniquely identify the client and server ($ID_c$ and $ID_s$, respectively). This message can optionally contain a set of preferences ($Preferences$). A preference set consists of graphs specified by the client that provide the server with an indication of what types of terms the user considers to be less desirable. These preferences are only present in the second round proposal, $Proposal_2$.

$Accept$ An accept message for round $j$ indicates that all of the terms of $Proposal_j$ have been accepted and the negotiation should terminate. The contents of this message constitute a privacy contract between the client and server. Accept messages are approval tokens that indicate in an authentic and non-repudiable way the acceptance of the last proposal policy. This token consists of a digital signature over the contents of $Proposal_j$ along with a time stamp.

$Reject$ The reject message indicates that the current negotiation has failed. This message is only sent by the client, and only at the end of the third round if the client does not accept any of the terms in the server's final proposal.

The rounds of an OBO negotiation proceed as follows.

**Round 1** The client initiates the first round by connecting to the server and requesting the server's privacy policy. The

server replies by sending the default policy, $P_1$, in the message $Proposal_1$. If the client accepts the default policy, it sends $Accept_1$, otherwise, Round 2 begins.

**Round 2** If the client rejects any term in the server's default policy from Round 1, then the client sends the message $Proposal_2$, including the client's preference set. If the server accepts the proposal, then it sends $Accept_2$, otherwise, Round 3 begins.

**Round 3** If the server rejects the client's counter-offer, it has one more chance for the negotiation to succeed. The server uses the client's preference set, and its own preferences, to create a best-offer policy, $P_3$, which is sent in the message $Proposal_3$. If the policy is acceptable, the client sends $Accept_3$, otherwise, the client sends $Reject$.

Throughout the remainder of this paper, we present a running example of an OBO negotiation between a client (Alice) and an online merchant (Bob). The negotiation reconciles Alice's privacy preferences with Bob's data collection. Bob obtains revenue through selling goods, as well as occasionally offering information about his customers to "partners". In addition, if Alice consents, Bob can use information about Alice to customize her experience at the site, and occasionally makes parts of her profile available for others to view. Once the preference model is formally specified, the running example shows how Alice and Bob specify their privacy preferences. Then, a three round negotiation example is given where Alice negotiates a contract with Bob for how he will handle her information such as her physical address, purchase information, and financial data.

## 4 Policies

Privacy policies are composed of *data elements* and *practice tags*. A *data element* is a reference to a single specific piece of information about an individual (e.g., a telephone number). Data elements are organized hierarchically into *data categories* and *data sets*, which can be used to refer to groups of data elements. We use the industry standard elements and categories defined by the W3C. For example, the data category "physical" contains all of the data elements that would allow someone to contact or locate an individual in the physical world. Here, we also define a set $AllData$, which contains every data element that applies to an individual. It is important to note that actual data about individuals does not occur inside of privacy policies, only labels that refer to pieces of information that the site may collect. For example, the policy may contain the token "telephone", but never an actual client telephone number.

In addition to declaring the types of data that they collect, entities must also be able to specify how they will treat that data. To do this, privacy policies associate *practice tags* with data elements. There are three types of practice tags: *recipients*, *retention*, and *purpose*. Recipients tags specify the parties that will have access to the data, retention tags specify how long the data will be stored, and purpose tags specify the ways in which the data will be used. Three disjoint sets, $RecipientTags$, $RetentionTags$, and $PurposeTags$ are defined that contain all supported recipients, retention and purpose tags, respectively. In this work we populate these sets with the tags based on those found in the P3P specification [7].

A privacy policy combines these atomic constituents as a collection of terms, organized into statements. In order to formulate privacy policies and preferences, it is often necessary to refer to the individual terms of a policy. These terms can be expressed in different ways. Using natural language, for example, one might formulate the following term: "we share your address with our shipping partners." Terms can also be expressed as formalized constructs in a machine readable format using P3P. Formally, a policy is a set of statements, that is $P = \{S_1, ..., S_n\}$, where each $S_i$ is a tuple of the form $S_i = (D_i, Rec_i, Ret_i, Pur_i)$, where $D_i \subseteq AllData$, $Rec_i \subseteq RecipientTags$, $Ret_i \subseteq RetentionTags$, and $Pur_i \subseteq PurposeTags$.

The statements of a policy can further be decomposed into terms, each of which is a tuple $T_i^X = (D_i, X_i)$, where $D_i$ is the set of data items specified in $S_i$ and $X$ indicates the term type and is one of: $Rec$, $Ret$, or $Pur$, with $X_i$ being the set of the appropriate type, also from $S_i$. This means that each statement consists of three terms, one for each type. Figure 2 shows how an example policy term might be represented formally. In this example, the P3P practice tags "ours", "delivery", "same", "others", and "public" are applied to the named set of data elements "physical", in order to provide the same semantics as the natural language statement. The P3P specification details the meaning of each tag [7].

Each OBO policy negotiation can be thought of as a set of synchronized concurrent negotiations, one for every term in the policy. This is because different sets of data elements have distinct preferences applied to them, and because it is impractical to directly compare the utility of practice tags of different types. The decisions and proposals made during the negotiation of one term in the policy do not affect the others. Therefore, the remaining discussion on policies will focus on how to analyze and interpret individual terms. In order to simplify the presentation here and without loss of generality, we will assume that all the terms given in the remainder of this paper refer to the same data set and are of the same data type. Using this assumption, we will treat references to $T = T_i^X = (D_i, X_i)$ as references to $X_i$.

**English:** "We share your address with partners who will use it to carry out delivery and perhaps in other ways as well. Your address may also be shared with other organizations, who's privacy policies are known to us, though they may differ from ours. It may also be shared with other site visitors, when appropriate."

**Formal Term:** $T^{Rec} = (\text{physical}, \{\text{ ours, delivery, same, others, public}\})$

**Figure 2. A term from Bob's default policy.**

# 5 Preferences

Agents must be able to reason about the relative quality of the terms that will be evaluated during a negotiation. A preference model defines the preferences of clients and servers. From these models, utility functions are derived that allow for the comparison and ranking of privacy policy terms. The models and utility functions must be defined explicitly and unambiguously as they are central to the functioning of the protocol and necessary in order to prove that a solution is Pareto-optimal.

## 5.1 Utility Functions

A cardinal utility function is a function of the form $U_C(T) \rightarrow \Re$ which maps a term to a real value, called the utility of $T$. In practice, it is unnecessary to completely specify such a utility function. The actual real-valued utilities are inconsequential, as long as policies can be sorted. To accomplish this, ordinal utility functions, $U_O(T_i, T_j)$, are defined for use by the software agents. These functions act as comparators that can be used to sort terms in non-descending utility order, without requiring the cardinal utility values. Ordinal utility functions can be easier to define, as only proportionality, and not magnitude is required.

**Definition 1.** *$U_O$ is an ordinal utility function if $\forall T_i, T_j$:*

$$U_O(T_i, T_j) = \begin{cases} 0 & \text{if } U_C(T_i) = U_C(T_j) \\ 1 & \text{if } U_C(T_i) > U_C(T_j) \\ -1 & \text{if } U_C(T_i) < U_C(T_j) \end{cases}$$

Because the number of terms that can be created with a finite set of tags is also finite, there must be at least one term that has utility greater than or equal to all other terms. The value of $U_C$ for this term is the upper bound on the range for that function and is called $MAX\_U$. The exact value of $MAX\_U$ is unimportant, as the term $T$ for which $U_C(T) = MAX\_U$ can be found using $U_O$.

Finally, each party specifies a threshold utility value $FAILURE\_U$ that determines the minimal threshold for terms it will accept.

## 5.2 Client Preference Model

The formulation of client privacy preferences follows a data-centric model [10]. All data categories and (as required) data elements are assigned three separate DAGs (directed acyclic graphs), one each for the *retention*, *recipients* and *purpose* tag types. These DAGs define partial orderings over tags that can be found in policy terms, with each tag occupying a node in the graph. For example, the recipients graph, $G^{Rec}$, gives a partial ordering over all tags in $RecipientTags$.

In this ordering, for tags $X$ and $Y$, $X \prec_G Y$ if there is a non-empty path in the graph from $X$ to $Y$. Also, $X \preceq_G Y$ indicates that either $X = Y$ (they are the same tag) or $X \prec_G Y$. We say that, $X$ and $Y$ are *independent* if neither $X \preceq_G Y$ nor $Y \preceq_G X$. In general we say that the preferability of a node is inversely related to this ordering, so that if $X \prec_G Y$, then the client prefers tag $X$ to tag $Y$.

Once graphs have been assigned to data elements, two sets of nodes, $A$ and $C$, in each graph are selected as acceptable and unacceptable cutoff frontiers, respectively. These frontiers must partition the graph's set of nodes, $N$, into three disjoint sub-sets: $Ideal$, $Acc$, and $Unacc$, which are defined as shown here:

$$\begin{aligned} Ideal &= \{n \in N \mid \exists a \in A : n \prec_G a\} \\ Acc &= \{n \in N \mid \exists a \in A, c \in C : a \preceq_G n \preceq_G c\} \\ Unacc &= \{n \in N \mid \exists c \in C : c \prec_G n\} \end{aligned}$$

These sets contain tags that the client considers to be ideal, acceptable and unacceptable, respectively. Ideal tags are those that the user "doesn't mind," that is, they have no negative impact on the utility of the policy as far as the client is concerned. Acceptable tags are those which the user would prefer not be included in the policy, but that are tolerable if unavoidable. Unacceptable tags are deal breakers. By placing a tag in this set, the client conveys that a negotiation should fail before the agent accepts a policy containing that tag. Users must be solicited for information about their tolerances in order to determine which of these sets each tag should belong to. This could be done in a guided fashion, with the system using the preference DAGs to selectively query the user about individual tag memberships until the borders are determined. Another approach would be to provide the user with some way to group or label the nodes free form, after appropriate instruction.

Once the graphs and cutoff frontiers are defined, the preferences for each data element $D_i$ are expressed by the tuple: $(D_i, G_i^{Ret}, G_i^{Rec}, G_i^{Pur}, A_i^{Ret}, C_i^{Ret}, A_i^{Rec}, C_i^{Rec}, A_i^{Pur}, C_i^{Pur})$, where the $G^X$'s are the preference DAGs, and the $A^X$'s and $C^X$'s are the acceptable and unacceptable cutoff frontiers for each graph. Data elements can be grouped together under the same set of preferences as desired.

From the specification of the client's preference model, it is possible to derive a utility function over terms for the client. Again, we assume that all terms and graphs refer to the same data sets and are of the same type. First, we define the concept of the least-preferred nodes in a set of tags.

**Definition 2.** *Given a preference graph $G$ and a term, $T$, the least-preferred nodes in $T$ are those in the set*

$$L(T, G) = \{x \mid x \in T \wedge \forall y \in T, x \not\prec_G y\}$$

In general, we say that the utility of the term as a whole is determined by the least preferable tags contained in that term, and is inversely proportional to the ordering over tags defined in the client's preference graph, $G$. This means that for tags $M$ and $N$, if $M \prec_G N$, and term $T_M$ contains tag $M$ but not tag $N$ and $L(T_M, G) = \{M\}$, $T_N$ contains tag $N$ but not tag $M$ and $L(T_N, G) = \{N\}$ and $T_{MN}$ contains both $M$ and $N$ and $L(T_{MN}, G) = \{N\}$, then $U_C(T_N) \leq U_C(T_M)$ and $U_C(T_N) = U_C(T_{MN})$. In addition, the following constraints apply:

1. $M \in \mathit{Ideal} \implies U_C(T_M) = \mathit{MAX\_U}$
2. $M \in \mathit{Unacc} \implies U_C(T_M) < \mathit{FAILURE\_U}$
3. $M, N \in \mathit{Acc} \wedge M \prec_G N \implies U_C(T_M) > U_C(T_N)$
4. $M, N \in \mathit{Acc} \wedge M = N \implies U_C(T_M) = U_C(T_N)$
5. $M, N \in \mathit{Acc} \wedge (M \not\preceq_G N \wedge N \not\preceq_G M) \implies U_C(T_M) = U_C(T_N)$

Given these constraints on $U_C$, we may now define the client's ordinal utility function over terms. This function formalizes the constraints listed above, generalizing them to apply to the case where nodes $M$ and $N$ are replaced with arbitrary sets of least-preferred nodes. Recall that $\mathit{Ideal}$, $\mathit{Acc}$, and $\mathit{Unacc}$ form a mutually exclusive partition.

**Definition 3.** *The client's ordinal utility function over terms with respect to graph $G$ is:*

$$U_O(T_i, T_j) = \begin{cases} 0 & \text{if } (L_i \subseteq \mathit{Ideal} \wedge L_j \subseteq \mathit{Ideal}) \vee \\ & (\mid F_j \mid = \mid F_i \mid \wedge \mid L_j \mid = \mid L_i \mid) \vee \\ & (L_i \cap \mathit{Unacc} \neq \emptyset \wedge \\ & L_j \cap \mathit{Unacc} \neq \emptyset) \\ 1 & \text{if } (L_i \cap \mathit{Unacc} = \emptyset \wedge \\ & L_j \cap \mathit{Unacc} \neq \emptyset) \vee \\ & (L_i \cap \mathit{Acc} \neq \emptyset \wedge L_j \cap \mathit{Acc} \neq \emptyset \wedge \\ & (\mid F_j \mid > \mid F_i \mid \vee \mid L_j \mid > \mid L_i \mid)) \\ -1 & \text{if } U_O(T_j, T_i) = 1 \end{cases}$$

*where $T_i$ and $T_j$ are the terms to be compared and $L_i = \{t \mid t \in L(T_i, G)\}$, $F_i = \{t \mid t \in L_i \wedge \exists u \in L_j \text{ s.t. } u \prec_G t\}$, and $L_j$ and $F_j$ are defined similarly for $T_j$.*

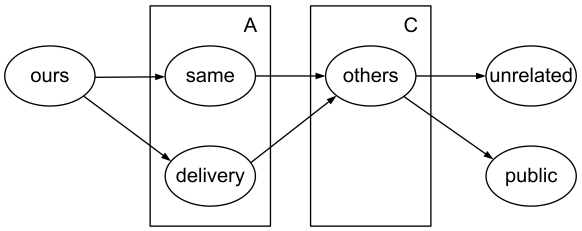


**Figure 3. Alice's recipients preference graph, with $A$ and $C$ cutoff frontiers.**

In other words, two terms have equivalent utility if all of their least-preferred nodes are in $\mathit{Ideal}$. If both terms have a least-preferred node in $\mathit{Unacc}$, or if both least-preferred tag sets contain the same number of tags that are less preferred than least-preferred tags in the other, and both least-preferred sets are the same size. A term that does not have any of its least-preferred nodes in $\mathit{Unacc}$ has greater utility than one that does. A term that has all of its least preferred nodes in $\mathit{Ideal}$ has greater utility than one that does not. Also, if both sets have least-preferred tags in $\mathit{Acc}$, then the one that has the most least-preferred tags that are less preferred than the other term's, or has fewer least-preferred tags in general, has less utility than the other.

With this function, we may identify terms with maximal utility, and those that have utility less than $\mathit{FAILURE\_U}$.

**Definition 4.** *The maximal utility term for the client is any term, $T$, that contains only ideal tags:*

$$U_C(T) = \mathit{MAX\_U} \iff T \subseteq \mathit{Ideal}.$$

**Definition 5.** *A client would rather a negotiation fail than accept any term $T$ that contains an unacceptable tag:*

$$U_C(T) < \mathit{FAILURE\_U} \iff T \cap \mathit{Unacc} \neq \emptyset.$$

**Running Example 1** (Alice's Preferences)**.** *Alice defines preference graphs to safeguard her personal data. She might compose these graphs herself using a software tool, select them from a pre-packaged source, or provide them as part of a security suite. Figure 3 shows one of these graphs that gives an ordering over recipient tags. Many possible configurations exist for each graph type. Alice might choose only one graph of each type for all her data, or apply different graphs to different data groups.*

*Alice groups her data as follows:*

$$D_1\text{(sensitive data)} = \{physical, purchase, financial\}$$

$$D_2\text{ (less sensitive data)} = \{e \mid e \in \mathit{AllData} \wedge e \notin D_1\}$$

*These sets contain labels of data elements, or categories of elements. Each group is assigned a set of preferences. This*

*means, for example, that if a privacy policy statement mentions any of the categories (e.g. physical), or members of the categories (e.g., address) in $D_1$, a certain set of preferences needs to be applied to that statement. In an actual configuration, each group is assigned a retention, recipients and purpose graph. For simplicity, in this example we only specify that Group $D_1$ is assigned the recipients graph shown in Figure 3. The cutoff nodes for this graph are $A = \{same, delivery\}$ and $C = \{others\}$. These cutoff nodes indicate that Alice has no problem with the server sharing her sensitive data with organizations that only use it for fulfilling her requests (ours), but is hesitant about the server sharing it with other organizations that might use it in other ways (same, delivery, others). Alice also does not want her sensitive information shared with the public or with organizations that have unknown policies (unrelated).*

### 5.3 Server Preference Model

The server side preference model is much simpler, this is the result of several factors. First, clients use the web for many distinct purposes, from shopping to web-mail to research. In contrast to clients, servers execute a relatively limited set of functions, all of which are governed by a given purpose or business model. The server preferences, therefore, are much more static then client preferences and are determined by the function of the server and the business requirements for the site collecting data from visitors. Also, since the object of the negotiation is the personal data of the clients, of which there are many, the client's preference model naturally needs more granularity and flexibility than the servers. This being the case, the server model groups preferences into just 2 categories:

1. *Req* - Because of technical or business model constraints, these terms must be in the final policy or the negotiation will fail.
2. *Pref* - These terms should be included in the final policy if possible, but if they cannot be, negotiation can still succeed.

We call members of *Req* required and members of *Pref* preferred.

Each of these categories is a set of pairs of the form $Category = \{K_1, ...K_m\}$ where each $K_i$ is a pair of the form $(D_i, t_i)$ where $D_i$ is a set of data elements and $t_i$ is a preference tag. Again, to simplify notation, we assume in what follows that all references to these categories refer to a single term and the data set $D$ and are all of the same type $X$. Therefore, we treat references to $Category$ as though it where the set of tags: $\{t \mid \exists K = (D, t) \wedge K \in Category\}$.

The server's utility function is much simpler than the client's. First, the server may not accept any term which does not contain all of its required tags. Also, for any term $T$ which contains all of the server's required tags, $U_C(T)$ is proportional to the number of preferred tags contained in the term. These constraints make it possible to fully specify the server's ordinal utility function over terms.

**Definition 6.** *The ordinal utility function over server terms:*

$$U_O(T_i, T_j) = \begin{cases} 0 & \text{if } (\mid R_i \mid = \mid R_j \mid) \wedge (\mid P_i \mid = \mid P_j \mid) \\ 1 & \text{if } (\mid R_i \mid > \mid R_j \mid) \vee \\ & ((\mid R_i \mid = \mid R_j \mid) \wedge (\mid P_i \mid > \mid P_j \mid)) \\ -1 & \text{if } U_O(T_j, T_i) = 1 \end{cases}$$

*where $T_i$ and $T_j$ are the terms to be compared, Req and Pref are the server's preference sets, $R_i = \{x \mid x \in T_i \wedge x \in Req\}$, $P_i = \{x \mid x \in T_i \wedge x \in Pref\}$, and $R_j$ and $P_j$ are defined similarly for $T_j$.*

That is, two terms have equivalent utility if they contain the same number of required tags and the same number of preferred tags. If two terms have different numbers of required tags, the term with more required tags has higher utility. Also, if two terms contain the same number of required tags, the one with the highest number of preferred tags has the greatest utility.

With the server utility function, the terms with maximal utility, and those that have utility less than $FAILURE\_U$ for the server can be identified.

**Definition 7.** *The maximal utility term for the server is any term, $T$, that contains all required and all preferred tags:*

$$U_C(T) = MAX\_U \iff Req \subseteq T \wedge Pref \subseteq T.$$

**Definition 8.** *A server would rather a negotiation fail than accept any term $T$ that does not contain all required tags:*

$$U_C(T) < FAILURE\_U \iff Req \nsubseteq T.$$

**Running Example 2** (Bob's Preferences)**.** *This is the portion of Bob's preferences that relates to recipients tags as applied to his customer's sensitive data:*

**physical, purchase and financial:**

**Required:** *delivery, others*

**Preferred:** *ours, same, public*

*These preferences mean that Bob must be able to give Alice's physical, purchase and financial information, if collected, to entities that will use it for delivery and, potentially, other purposes (delivery). He also must be allowed to share it with companies accountable to him, but who may have privacy policies that he is not familiar with (others). Bob would also like the option of sharing that data with organizations that only use it to help fulfill any orders placed by the user (ours), and partners having similar privacy policies (same). Finally, he would prefer having the option to share it with other visitors, when appropriate (public).*

| *Agent* | *Agent task* | *Preference constraints* | *Protocol constraints* |
|---|---|---|---|
| Client | Accept proposal | Reject policies containing unacceptable tags | None |
| Client | Counter-proposal | Remove all unacceptable nodes | Term should have highest possible utility and only contain $Ideal$ tags |
| Server | Accept proposal | Only accept policies containing all the server's required tags | None |
| Server | "Best offer" | Ensure that the policies contains all required tags and as many preferred tags as possible | Server not decrease client utility any more than necessary for the negotiation to succeed |

**Table 1. Constraints on agent behavior during OBO negotiations.**

## 6 Negotiation Strategy

Agents are constrained in the formulation of proposal policies in that they must follow strategies that are consistent with the preferences of the party they represent, while at the same time fulfilling the guidelines specified by the protocol. Table 1 outlines these constraints. Any agent that acts within these constraints can engage in OBO negotiations. However, not all strategies that are consistent with these constraints are guaranteed to be fair (produce Pareto-optimal policies). Here we describe a set of rules that meet these constraints and that, when followed by both parties, are sufficient to always produce Pareto-optimal results. This set of rules is the "OBO Pareto-optimal strategy".

**Rule 1** (Initial Offer Rule)**.** *The server's initial offer term is* $T = Req \cup Pref$.

**Rule 2** (Early Acceptance Rule)**.** *In rounds 1 and 2, a party,* $A$*, may only accept a proposal term* $T$ *from party* $B$ *if* $U_C^A(T) \geq U_C^A(T')$*, where* $T'$ *is the counter-proposal term that* $A$ *would send to* $B$ *upon rejection of* $T$.

**Rule 3** (Client Counter-proposal Rule)**.** *Given an initial proposal term* $T$ *from the server and client preference graph* $G$*, the client formulates a new term* $T' = \{t \mid t \in T \wedge t \in Ideal$ *according to the* $A$ *cutoff frontier for* $G\}$.

**Rule 4** (Server Best-offer Rule)**.** *Given a proposal term* $T$ *from the client, the server formulates its best-offer term* $T''$ *in two stages. First, the server inserts all of its* Req *tags into the term, creating a new set* $T' = T_i \cup Req$*. Next, it adds all of its preferred tags into the set that it can, without decreasing the utility of the term for the client by creating a new set* $T'' = T' \cup \{t \mid t \in Pref \wedge \exists s \in T' \text{ s.t. } t \prec s\}$.

**Rule 5** (Client Final Acceptance Rule)**.** *Given a best-offer proposal term* $T$ *and client preference graph* $G$*, the client accepts the term only if* $T \cap Unacc = \emptyset$ *according to the* $C$ *cutoff frontier of* $G$ *and rejects otherwise.*

**Running Example 3** (Alice and Bob Negotiate)**.** *Based on their preferences, Alice and Bob apply the rules of the Pareto-optimal strategy in the negotiation over the recipients of Alice's address information as follows. In Round 1, Bob sends this term (Rule 1):*

$$T_1^{Rec} = (D_1, \{\ ours,\ delivery,\ same,\ others, public\})$$

*Alice rejects Bob's offer (Rules 2 and 3), sends her preference graphs, data groupings, and the following policy (Rule 3):*

$$T_1^{Rec} = (D_1, \{\ ours\})$$

*This counter-proposal decreases the number of recipients with which Alice's sensitive information can be shared. In the final round Bob rejects Alice's proposal (Rules 2 and 4) and formulates a "best offer" policy (Rule 4):*

$$T_1^{Rec} = (D_1, \{\ ours,\ delivery,\ same,\ others\})$$

*This term re-introduces Bob's* required *tags delivery and others that were removed by Alice. Also, the* preferred *term* same *was re-introduced because* $\mathbf{same} \prec_G \mathbf{others}$*. Finally, Alice accepts the policy (Rule 5).*

## 7 Protocol Evaluation

The OBO protocol is complete by definition; all negotiations are guaranteed to terminate within three rounds. The protocol is also fair and secure. Fairness is evaluated by proving that terms resulting from a successful OBO negotiation are Pareto-optimal. The security of the protocol is analyzed as well, using a threat model to identify potential problems in the security of the protocol, and then presenting implementation design considerations that could mitigate these problems.

### 7.1 Fairness Analysis

Pareto-optimality, or Pareto-efficiency is a property of some game and negotiation end-states. It is often used as

an indication that the benefits of successful negotiations are balanced for both parties [2, 9, 4]. For a state to be Pareto-optimal, it must be the case that there is no other state that is better for all parties in the negotiation, or better for at least one party and not worse for all the others.

**Definition 9.** *Given two negotiating parties $P_1$ and $P_2$, a policy term $T$ is Pareto-optimal if for all other $T'$ the following holds:*

$$((U_C^{P_1}(T') = U_C^{P_1}(T)) \wedge (U_C^{P_2}(T') = U_C^{P_2}(T))) \vee$$
$$((U_C^{P_1}(T') < U_C^{P_1}(T)) \vee (U_C^{P_2}(T') < U_C^{P_2}(T))).$$

Recall that the cardinal utility function, $U_C$, for a negotiating party is implicitly defined by a corresponding ordinal utility function, $U_O$, that effectively orders any two related terms in a negotiation (see Definition 1). A Pareto-optimal term is thus a term for which all other related terms are less desirable for one negotiating party or have the same utility for both negotiating parties according to their respectively defined ordinal utility functions.

**Theorem 1.** *If the parties in an OBO negotiation both follow the OBO Pareto-optimal strategy in Section 6, then a successful negotiation always produces a Pareto-optimal term. The proof of Pareto optimality is given in [8].*

### 7.2 Security Analysis

Negotiators must keep certain information secret in order to maintain fairness. Client's must hide the $A$ and $C$ node sets for each graph from the server. Server's must hide whether a tag is required or preferred. If the server knows the set $C$ for a given graph, it can add as many of its preferred tags as it would like, up to and including the members of $C$, meaning that it has no reason to make an effort to meet the client's preferences as closely as possible.

Clients and servers are both vulnerable to probing attacks if two parties engage in mulitple OBO negotiations with each other. If one party is stingy and gradually reveals more information across failed negotiations, the point at which a negotiation finally succeeds can leak information about the other parties' preferences. A client can avoid this by maintaining a cache of negotiated policies to avoid unnecessary renegotiations. Servers are less able to track negotiations with a single client, but there is less risk to such attacks since server preferences are usually publicized in a server's privacy policy.

## 8 Conclusions and Future Work

The Or Best Offer privacy policy negotiation protocol is complete, fair and secure. Its formal underpinnings provide properties not found in prior negotiation protocols. It is backwards-compatible with P3P.

A significant contribution of this work is the novel graphical model for expressing client privacy preferences and utility functions, derived from preference models, that allow for the comparison of policy terms. In addition, the definition of utility functions allows for the application of game theoretical concepts to analyze the properties of the protocol. This formalism allows conjecture about alternative negotiation strategies and algorithms. As new strategies are envisioned, fairly simple analysis using concepts such as Pareto-optimality and Nash equilibrium would yield an understanding of their potential performance.

The graphical model may be an improvement over rule-based preference models in terms of usability. Other future work includes increasing the expressiveness of the client and server prefernce models.

## References


[1] M. Bennicke and P. Langendorfer. Towards automatic negotiation of privacy contracts for internet services. In *11th IEEE International Conference on Networks*, Sydney, Australia, October 2003.

[2] R. Lau. Adaptive negotiation agents for e-business. In *Proceedings of the 7th international Conference on Electronic Commerce*, Xi'an, China, August 2005.

[3] M. Maaser and P. Langendoerfer. Automated negotiation of privacy contracts. In *29th Annual International Computer Software and Applications Conference (COMPSAC'05)*, July 2005.

[4] V. Robu, D. Somefun, and J. L. Poutré. Modeling complex multi-issue negotiations using utility graphs. In *Proceedings of the 4th international joint Conference on Autonomous Agents and Multiagent Systems*, Utrecht, Netherlands, July 2005.

[5] S. Spiekermann, J. Grossklags, and B. Berendt. E-privacy in 2nd generation e-commerce: privacy preferences versus actual behavior. In *3rd ACM conference on Electronic Commerce*, Tampa, Florida, October 2001.

[6] R. Thibadeau. Privacy server protocol: Short summary. http://yuan.ecom.cmu.edu/psp/SummaryInterop.pdf, November 2000.

[7] W3C. The platform for privacy preferences 1.1 (P3P1.1) specification. http://www.w3.org/TR/2005/WD-P3P11-20050701/, 2005.

[8] D. Walker. Or best offer: A privacy policy negotiation protocol. Master's thesis, Brigham Young University, June 2007.

[9] S.-H. Wu and V.-W. Soo. Game theoretic reasoning in multi-agent coordination by negotiation with a trusted third party. In *Proceedings of the 3rd international Conference on Autonomous Agents*, Seattle, Washington, May 1999.

[10] T. Yu, N. Li, and A. I. Anton. A formal semantics for p3p. In *ACM Workshop on Secure Web Services*, Fairfax, VA, October 2004. ACM Press.