OR BEST OFFER: A PRIVACY POLICY NEGOTIATION PROTOCOL

by

Daniel D. Walker IV

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computer Science

Brigham Young University

August 2007

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Daniel D. Walker IV

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Kent E. Seamons, Chair
Date	Eric Mercer
Date	Parris K. Egbert

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Daniel D. Walker IV in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

Kent E. Seamons
Chair, Graduate Committee

Accepted for the Department

Parris K. Egbert
Graduate Coordinator

Accepted for the College

Thomas W. Sederberg
Associate Dean, College of Physical and Mathematical Sciences

ABSTRACT

OR BEST OFFER: A PRIVACY POLICY NEGOTIATION PROTOCOL

Daniel D. Walker IV

Department of Computer Science

Master of Science

Users today are concerned about how their information is collected, stored and used by Internet sites. Privacy policy languages, such as the Platform for Privacy Preferences (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 when evaluating these privacy policies. This approach is inflexible and gives the server ultimate control over the privacy of web transactions. 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 parties. That is, it remains fair to both the client and the server.

ACKNOWLEDGMENTS

I would like to thank my family for all they have done to help me achieve my goals. Most importantly, this work would not have been possible without the love and support of my beautiful wife Choka, and my children Zayaa and Daniel V, whose smiles and hugs made all of this bearable.

I am also very grateful to my advisor, Dr. Kent Seamons, for the many hours that he has spent proofing and helping to edit this this work for publication and for all of his guidance and advice. I am also grateful to Dr. Eric Mercer, who went above and beyond his duties as a second committee member and greatly contributed to the formalization of the theory presented here and the development of the work.

I also acknowledge all of the ISRL members whose feedback and support helped develop this idea into a thesis.

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.

Table of Contents

List of Tables

List of Figures

Chapter 1 — Introduction

Reports of identity theft and the loss and misuse of personal information fuel increased privacy concerns for many Internet users, who worry about the personal information that websites collect. To help alleviate these concerns, many websites publicize their privacy practices.

One way that privacy policies may be published is through the use of the Platform for Privacy Preferences (P3P) [23], an XML language designed to specify sites intend to handle information they collect about visitors to their site. Usually, a site publishes its P3P policy in a well-known location on its server. When a client visits the site, a software agent 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. If the agent determines that the policy is incompatible with the user's preferences, the transaction is discontinued or limited, optionally with a message to the user explaining the nature of the incompatibility.

This "take it or leave it" approach is static and confining given the dynamic nature of the Internet and the flexibility of user preferences, which depend on context and other factors. Spiekermann et al. [20] 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 that information

in rather promiscuous ways, it may be willing to collect less information, and use that information in more protected ways, but only if a user specifically requests such protections.

One way to increase the flexibility of P3P is by introducing per-session privacy negotiations that are carried out by software agents according to an established protocol [4, 14, 21]. Their purpose is to produce fine-grained privacy contracts that dictate the contents of the privacy policy that governs the use of data revealed or collected during a single transaction.

Before it becomes practical and effective to implement per-session privacy policy negotiations, several developments need to occur. First there must be a way to guarantee the authenticity of the resulting policies, and to verify which parties negotiated to produce them. Second, mechanisms must be put in place to help enterprises manage and enforce these policies. Third, some way must be found to detect policy violations. Fourth, legal measures must exist to offer recourse when violations occur. Finally, appropriate protocols for negotiating the terms of the privacy policies must be formulated. This paper seeks to address only this final requirement, the formulation of a privacy policy negotiation protocol. We assume the existence of a PKI infrastructure as a central part of the protocol to provide the needed authenticity. The problems of developing enforcement mechanisms, violation detection techniques, and the appropriate legal constructs, though important prerequisites for the deployment of the protocol, are beyond the scope of this work.

Other privacy policy negotiation protocols have been developed [4, 14, 21]. 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.

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) style of negotiation, similar to sellers who advertise an item for a fixed price and then express a willingness to entertain a "best offer" in order to learn what buyers might be willing to pay. 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.

The remainder of this document is organized as follows: Chapter 2 introduces related work. Chapter 3 gives a high level explanation of the protocol along with definitions of each message type used and a specification for when each message is sent. Chapter 4 introduces a running example that will be referenced to explain various concepts throughout the remainder of the document . Chapter 5 defines the policy model used in OBO. In Chapter 6 the problem of reasoning about privacy policies is addressed through the introduction of preference models and utility functions used by agents conducting OBO negotiations. The negotiation strategy employed by negotiating agents is given in Chapter 7. Chapter 8 evaluates how well OBO meets its goals, including a proof showing the Pareto-optimality of OBO negotiations. Finally,

Chapter 9 contains conclusions and future work.

Chapter 2 — Related Work

The idea of negotiating per-session privacy policies was considered for inclusion in the P3P specification. The idea was rejected, however, when the platform's designers were unable to envision scenarios in which this capability would be useful [5]. Since then, there have been two proposals for per-session privacy policy negotiation protocols.

The first such proposal was made by Bennicke and Langendorfer [4]. In their protocol, a negotiation is a process by which a privacy contract is proposed and then incrementally modified to meet the demands of both parties. Negotiators have demands that can either be mandatory or optional. The goal of a negotiating party is to have all of its own mandatory demands met and as many of its optional demands met as possible by the final contract. Each party alternatively assumes the roles, *proposal maker* and *acceptor*. The proposal maker proposes a contract that meets at least its own mandatory demands, and the accepter can then respond in one of several ways to refine the policy until both parties agree to accept it. In order to enable negotiations, Bennicke and Langendorfer introduce modifications to the P3P and APPEL (a rule-based P3P preference exchange language) specifications. A later paper, by Langendorfer and Maaser[14], includes a more detailed description of the algorithms used to resolve conflicting APPEL messages caused by multiple rule matches on the same policy.

The second negotiation protocol is the Privacy Server Protocol Project (PSP) [21], which 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. The assumption is that both parties may reveal sensitive data, and the client should

therefore agree to respect the server's privacy also. Despite its bilateral nature, PSP is very similar to the Bennicke-Langendorfer proposal, in that the protocol requires the client and server to exchange proposals and counter-proposals until one side finally agrees to the last proposal made by the other. It also uses a rule-based preference model based on APPEL.

One limitation of these privacy policy negotiation protocols is the fact that they are incomplete and could require a potentially infinite number of rounds. Moreover, these methods do not have models in place to inform negotiating agents as to how policy terms rank with regards to preference, so they may make concessions that are actually counter-productive to the negotiation.

Chapter 3 — OBO Protocol Specification

The goals of OBO are to be complete, fair, and secure. A protocol is complete if it always terminates in a successful or unsuccessful outcome. OBO is complete by definition and only admits three rounds of negation. A protocol is fair if it does not favor one party over the other. OBO is fair, and we prove its fairness using Pareto-optimality from game theory in Section 8.0.2. Pareto-optimality is the notion that no other solution exists that can better benefit either party participating in the game. In other words, in a successful negotiation, neither the client nor the server can better meet their own needs without causing the policy to be rejected by one or the other. A protocol is secure when it protects the negotiating parties or a third party from manipulating the negotiation process. OBO is secure assuming adequate legislation, PKI, etc. as discussed in the introduction. A full analysis of OBO security is contained in Section 8.0.2.

3.1 Overview

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 to the server. This counter proposal will remove any of the portions of the policy that the client does not feel completely comfortable with. With the counter-proposal, the client is effectively telling the server: "Give me this much privacy, or make me

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

Figure 3.1: OBO negotiation messages. Here, $sign(X)$ is a digital signature over X.

---

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.

Negotiations are carried out piecewise over distinct portions of the policy referred to as *terms*. In fact, an OBO policy negotiation can be thought of as 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 the negotiations fail, the overall negotiation fails to produce a consensus policy.

3.2 Message Definitions

There are three types of messages in the OBO protocol (see Figure 3.1).

Proposal The proposal message for round j ($Proposal_j$) contains the 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 desireable. These preferences are only present in the second round proposal, $Proposal_2$.

Accept An accept message for round j is used to indicate that all of the terms of $Proposal_j$ have been accepted and the negotiation should terminate. The contents of this message constitute the 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 t_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.

3.3 Message Flow

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 of the terms in the server's default policy from Round 1, then the client begins the second round of negotiation by sending the message $Proposal_2$, including the client's preference set. If the server decides to accept the policy proposal, then it sends $Accept_2$.

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 the best-offer policy, P_3, which is sent in the message $Proposal_3$. If the policy is acceptable, the client sends $Accept_3$. If the client does not approve of the final offer, then it sends a $Reject$ message.

Chapter 4 — Running Example

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.

Chapter 5 — Policies

Privacy policies are composed of the atomic constituents *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. In this work, we make use of the elements and categories defined by the W3C, as they are considered industry standards. For example, the data category "physical," as defined by the W3C, 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 [23].

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 5.1 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 [23].

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,

English: "We share your address with shipping partners who will use it to carry out delivery and may use it 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 be different than our own. It may also be shared with other site visitors, when appropriate."

Formal Term:

$$T^{Rec} = \quad (\text{physical}, \{\ \text{ours},\ \text{delivery},\ \text{same},\ \text{others}, \text{public}\})$$

Figure 5.1: A term from Bob's default policy.

---

we will treat references to $T = T_i^X = (D_i, X_i)$ as references to X_i.

Chapter 6 — Preferences

Agents must be able to reason about the relative quality of the terms that will be evaluated during a negotiation. To this end preference models are defined to encode the unique needs and preferences of individual clients and servers. From these models, utility functions are derived that allow for the comparison and ranking of privacy policy terms. It is important that these models and their corresponding utility functions be defined explicitly and unambiguously, as they are central to the functioning of the protocol and necessary in order to prove that a given solution is Pareto-optimal.

6.1 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 not necessary to completely formulate such a utility function, however. 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, because 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, it must be the case that there is 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 real value of MAX_U is unimportant, as the term T for which $U_C(T) = MAX\_U$ can be found using U_O.

Finally, for the purpose of conducting negotiations, each party must choose a threshold utility value $FAILURE\_U$ that determines the point at which that party would rather have a negotiation fail than accept a term with utility below that threshold.

6.2 Client Preference Model

The formulation of client privacy preferences follows a data-centric model [25]. 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:

$$Ideal = \{n \in N \mid \exists a \in A : n \prec_G a\}$$

$$Acc = \{n \in N \mid \exists a \in A, c \in C : a \preceq_G n \preceq_G c\}$$

$$Unacc = \{n \in N \mid \exists c \in C : c \prec_G n\}$$

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 (respectively) for each graph. Sets of 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. If $M \in \mathit{Ideal}$, then $U_C(T_M) = \mathit{MAX\_U}$
2. If $M \in \mathit{Unacc}$, then $U_C(T_M) < \mathit{FAILURE\_U}$
3. If $M, N \in \mathit{Acc}$ and $M \prec_G N$ then $U_C(T_M) > U_C(T_N)$
4. If $M, N \in \mathit{Acc}$ and $M = N$, then $U_C(T_M) = U_C(T_N)$
5. If $M, N \in \mathit{Acc}$ and M and N are independent ($X \not\preceq_G Y \wedge Y \not\preceq_G X$), then $U_C(T_M) = U_C(T_N)$

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 *Ideal*, *Acc*, and *Unacc* form a mutually exclusive partition on the term.

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

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

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.

In other words, two terms have equivalent utility if all of their least-preferred nodes are in *Ideal*. If both terms have a least-preferred node in *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 *Unacc* has greater utility than one that does. A term that has all of its least preferred nodes in *Ideal* has greater utility than one that does not. Also, if both sets have least-preferred tags in *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.

Based on this function, it is possible to identify terms with maximal utility, and

those that have utility less than $FAILURE\_U$.

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

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

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

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

6.2.1 Alice's Preferences

At some point, Alice has defined preference graphs to safeguard her personal data. Alice might have composed these graphs herself using a software tool. Another possibility is that she selected these graphs from some pre-packaged source, or had them provided as part of a security suite. Figure 6.1 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:

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

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

These sets contain labels of data elements, or categories of elements. No actual user data is included in these sets. 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

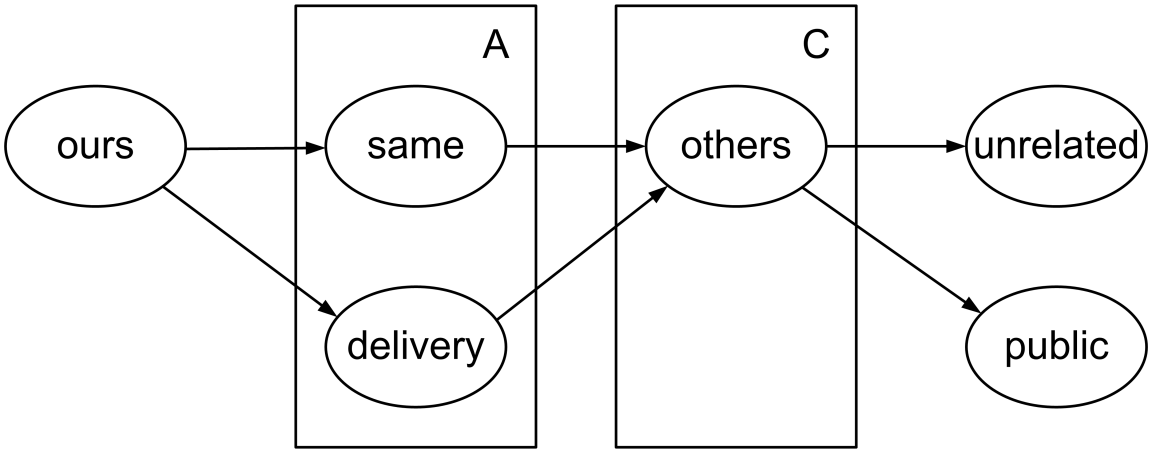

Figure 6.1: Alice's recipients preference graph, with A and C cutoff frontiers.

---

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 6.1. 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 privacy policies (unrelated).

6.3 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 terms for the server is:*

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

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:*

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

6.3.1 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 that are 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).

Chapter 7 — Negotiation Strategy

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 7.1: Constraints on agent behavior.

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 7.1 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* $T' = \{t \mid t \in T \wedge t \in Ideal \text{ according to the } A \text{ cutoff frontier for } G\}$*.*

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.*

7.1 Alice and Bob Negotiate

Based on their established 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):

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

Alice chooses not to accept (Rules 2 and 3), instead sending her preference graphs, data groupings (with an indication of which graphs apply to each), and the following policy (Rule 3):

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

This counter-proposal decreases the number of recipients with which Alice's sensitive information can be shared.

In the final round Bob does not accept Alice's proposal (Rules 2 and 4). Instead, he formulates the "best offer" policy shown here (Rule 4):

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

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 $same \prec_G others$. At this point, Alice accepts the policy and the negotiation succeeds (Rule 5).

Chapter 8 — Protocol Evaluation

As discussed in Chapter 3, a viable privacy policy negotiation protocol must be complete, fair and secure. The OBO protocol is complete by definition; all negotiations are guaranteed to terminate within the three rounds specified. This chapter shows that 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.

8.0.1 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 [13, 24, 19]. 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.

As usual, all terms referred to here are assumed to be related, meaning they are defined on a common data set and tag type. This means that we can treat references to $T = T_i^X = (D_i, X_i)$ as references to X_i. To enhance readability, we therefore refrain from writing out the entire contents of each term, and allow their constituent tag sets to stand in for the term itself in the discussion that follows. We are now ready to state the main theorem of the paper:

Theorem 1. *If the parties in an OBO negotiation both follow the OBO Pareto-optimal strategy defined in Chapter 7, then a successful negotiation always produces a Pareto-optimal term.*

Proof. There are three cases in which an OBO negotiation may succeed. Let T be a term produced by a successful negotiation and G be the corresponding client preference graph partitioned appropriately. We show that, in each of these cases, T is Pareto-optimal. We distinguish client and server utility functions by adding superscripts of C and S respectively.

1. Client Accepts Server's Initial Offer

   The term has maximal utility for the client and the server because the initial offer contains only ideal tags for the client and all of the required and preferred tags for the server (see Rules 1, 2, 3, and Definitions 4 and 7). Because both parties are at maximal utility, T is Pareto-optimal.

2. Server Accepts Client's Counter-Offer

   Assume for contradiction that T is not Pareto-optimal. By Rules 2 and 3, the counter offer contains strictly fewer tags than the initial offer and all of the tags

in the counter offer are ideal for the client. This means that $U_C^C(T) = MAX\_U$. If T is not Pareto-optimal, then

$$\exists t : U_C^C(T \cup \{t\}) \geq MAX\_U \wedge U_C^S(T \cup \{t\}) > U_C^S(T)$$

By Rules 2, and 4, T contains all of the server's required tags. The only tags that the server can add to T are therefore preferred tags removed from the initial proposal; however, adding any removed preferred tag decreases the utility of T for the client since these tags are not ideal (see Rule 3). Formally:

$$\forall t \in Pref \setminus T, t \notin Ideal \wedge U_C^S(T \cup \{t\}) >$$
$$U_C^S(T) \wedge U_C^C(T \cup \{t\}) < MAX\_U$$

This contradicts the assumption that T is not Pareto-optimal, since there is no tag that we can add to T that maintains the utility of the term for the client and increases the utility of the term for the server.

3. Client Accepts Server's Best Offer

   Again, for contradiction assume that T is not Pareto-optimal, meaning that either:

   $$\exists t : U_C^C(T \setminus \{t\}) > U_C^C(T) \wedge U_C^S(T \setminus \{t\}) \geq U_C^S(T)$$

   or

   $$\exists t : U_C^C(T \cup \{t\}) \geq U_C^C(T) \wedge U_C^S(T \cup \{t\}) > U_C^S(T)$$

   We will examine these statements in order.

   First, we know that there is some tag in both T and Acc such that

   $$U_C^C(T \setminus \{t\}) > U_C^C(T)$$

(by Definition 3), because otherwise the server would not have required an additional round by Rules 2 and 4. However, by Definition 3,

$$\forall t \in T, U_C^S(T \setminus \{t\}) < U_C^S(T)$$

meaning that the first statement is false.

Next, we also know from Rule 4 that T already contains all of the server's required tags, and all of the preferred tags that it could add without decreasing the utility for the client below that of the term $T' = \mathit{Req}$. So, any tag that the server could add to increase its own utility is either less preferred for the client than some tag already in the least-preferred set of T or it increases the size of the set of least-preferred tags. This means that

$$\forall t \in \mathit{Pref} \setminus T, U_C^S(T \cup \{t\}) > U_C^S(T) \wedge$$
$$U_C^C(T \cup \{t\}) < U_C^C(T)$$

by Definition 3, making the second statement false.

The fact that both statements are false contradicts the assumption that T is not Pareto-optimal.

□

8.0.2 Security Analysis

Each side in an OBO negotiation must keep certain information secret, so that the other cannot act in ways that are inconsistent with the intentions of the protocol. For the client, this means hiding the A and C node sets for each graph from the server. Conversely, the client should also not be able to ascertain whether each tag in a term is required or preferred from the server's point of view.

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.

The most obvious threat to either client or server is that of a probing attack, which can only be carried out if one party is able to convince the other party to engage in multiple instances of OBO negotiations. The attacker tries to determine the other side's secrets by carefully formulating its various proposals. For example, to discover the cutoff frontier C of a given client, a server might include in its best offer proposal only one tag that is least preferred according to the client's preference graph. If this offer is refused, the server knows that that tag is definitely a member of the unacceptable set of tags. The next time the client attempts to connect, the server picks a node that is the direct ancestor of the least preferred tag. In this fashion, the server ascertains, over a relatively small number of negotiations, every member of the unacceptable set and thus can know exactly what the user's cutoff nodes are for that graph. The fact that the client has an ordered graph, detailing the preference gradient, means that the server-side probing attack can be much more effective than it would be if the server had to use a brute-force technique to deduce C.

The risk of probing can be mitigated in implementations of the OBO protocol, however. For example, clients can keep a cache of negotiated privacy policies, indexed by the identity of the sites with which they were negotiated. Then, when visiting a site with which the client already has a relationship, it can refuse to engage in additional negotiations, insisting instead that the previously negotiated policy be used. This sort of caching system must be carefully implemented, however, as website data collection requirements may change over time as enterprise policies change, or as the relationship between the client and the enterprise changes. For example, corporate mergers may require websites to adopt new data collection and use policies. Also, a

user who has habitually visited an online commerce site without making purchases might need to accept the fact that the privacy contract between herself and the site must change in the event that she decides to actually buy something. This is because the relationship between herself and the site has changed and the site must now collect more information about her and share that information with other entities, such as a shipping company.

On the server side, probing attacks are harder to defend against for two reasons. First, the number of clients that connect to a given server can be much greater than the number of servers that any one client connects to. In addition, while many servers have distinct domain names and certificates issued by trusted third-parties to authenticate their identity, individual clients have no such unique identification mechanism, as clients can share or create new credentials. However, the ease with which probing attacks can be carried out against servers is offset by the relatively small gains that such attacks yield. The preferences of the server are already mostly public, as they will almost always be expressed in the natural language privacy policy of the server.

Chapter 9 — 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. The protocol is backwards-compatible with current P3P negotiation approaches.

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, such as the proof of Pareto-optimality in Chapter 8. This formalism allows conjecture about alternative negotiation strategies and algorithms for clients and servers. 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.

One interesting point of the protocol is that the messages in the first round can be replaced with a more standard privacy policy exchange procedure, like that used in P3P today. This would allow for the gradual deployment of OBO capable agents on the Internet, as OBO enabled agents can seamlessly interact with clients and servers that do not support OBO negotiations. Clients and servers that do not support OBO negotiations would not be able to distinguish OBO-enabled agents from any other entities. OBO-enabled servers could indicate their negotiation capabilities by embedding meta-data in their default policies.

The graphical model has the potential to offer an improvement over rule-based preference models in terms of usability, as graphical user interfaces could allow users or administrators to edit such models without any prior knowledge of the privacy

policy language syntax. A prototype implementation of such an interface can be built to evaluate the hypothesized improvement in usability.

An important area for future work is to increase the expressiveness of the client preference model. The current model is less expressive than a rule based system, for example. It precludes the ability to express higher-ordered preferences that involve complex interactions between tags of different types. For example, using a rule-based system it would be possible to express the preference "It is never acceptable for my address to be stored indefinitely, unless the collecting enterprise never shares it or sells it to anyone else." In the current OBO client preference model, this type of preference cannot be expressed.

Future work might also focus on producing a richer preference model for the server, which would permit the specification of more complex preferences, in which combinations of tags in conjunctions and disjunctions could be specified together as being required or preferred. To accomplish this, the server could send a list of alternative policies representing its best offer, one for each grouping in a disjunction. The client could select the alternative that it prefers most and respond with an acceptance message for the chosen alternative.

References

[1] Paul Ashley, Satoshi Hada, Gunter Karjoth, and Matthias Schunter. E-P3P privacy policies and privacy authorization. In *Proceedings of the Workshop on Privacy in the Electronic Society Washington*, Washington, D.C., November 2002.

[2] AT&T. AT&T privacy bird tour. http://privacybird.com/tour/1_2_beta/tour.html, 2003.

[3] Michael Backes, Gunter Karjoth, Walid Bagga, and Matthias Schunter. Efficient comparison of enterprise privacy policies. In *Proceedings of the ACM Symposium on Applied Computing*, Nicosia Cyprus, March 2004.

[4] 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.

[5] Lorrie Faith Cranor. *Web Privacy with P3P*. O'Reilly & Associates, Inc., Sebastopol, CA, first edition, 2002.

[6] Nicholas Fehlberg. P3P, cookies and ie6.0: A case study. March 2004.

[7] Simson Garfinkel and Lorrie Cranor. P3P: Privacy primer. http://www.oreillynet.com/pub/a/network/excerpt/p3p/p3p.html?page=1, February 2002.

[8] Simson Garfinkel and Gene Spafford. *Web security, privacy & commerce*. O'Reilly, second edition, 2001.

[9] H. Hochheiser. The platform for privacy preference as a social protocol: An examination within the U.S. policy context. *ACM Transactions on Internet Technology*, November 2002.

[10] Tatsuro Ichiishi. *Game Theory for Economic Analysis*. Academic Press, 1983.

[11] Gunter Karjoth, Matthias Schunter, and Els Van Herreweghen. Translating privacy practices into privacy promises – how to promise what you can keep. In *Proceedings of the 4th International workshop on policies for distributed systems and networks*, Como, Italy, June 2003.

[12] Gunter Karjoth, Matthias Schunter, and Michael Waidner. Platforms for enterprise privacy practices: privacy-enabled management of customer data. In *Proceedings of the 2nd workshop on privacy enhancing technologies*, San Francisco, California, April 2002. Springer Verlag.

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

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

[15] Microsoft. Microsoft P3P implementation in Internet Explorer 6.0 and Windows XP fact sheet.
http://www.microsoft.com/presspass/press/2001/mar01/privacytoolsiefs.asp, March 2001.

[16] Nua.com. Consumer internet barometer: More americans online, but trust still an issue.
http://www.nua.com/surveys/index.cgi?f=VS&art_id=905358466&rel=true, Oct 2002.

[17] Stephanie Olsen. Doubleclick seeks input on new policy.
http://news.com.com/DoubleClick+seeks+input+on+new+policy/2100-1023_3-267828.html?tag=st.rn, June 2004.

[18] Birgit Pfitzmann and Michael Waidner. Privacy in browser-based attribute exchange. In *Proceedings of the Workshop on Privacy in the Electronic Society*, Washington, D.C., November 2002. ACM press.

[19] Valentin Robu, D.J.A. Somefun, and J.A. La 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.

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

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

[22] W3C. A P3P preference exchange language 1.0 (APPEL 1.0).
http://www.w3.org/TR/P3P-preferences, 2002.

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

[24] Shih-Hung Wu and Von-Wun 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.

[25] Ting Yu, Ninghui Li, and Annie I. Anton. A formal semantics for P3P. In *ACM Workshop on Secure Web Services*, Fairfax, VA, October 2004. ACM Press.