Vision Paper: Enabling Privacy for the Paranoids

Gagan Aggarwal, Mayank Bawa, Prasanna Ganesan, Hector Garcia-Molina
Krishnaram Kenthapadi, Nina Mishra, Rajeev Motwani
Utkarsh Srivastava, Dilys Thomas, Jennifer Widom
Stanford University
Stanford, CA 94305
{*gagan,bawa,pganesan,hector,kngk,nmishra,rajeev,usriv,dilys,widom*}*@cs.stanford.edu*

Abstract

P3P [27, 32] is a set of standards that allow corporations to declare their privacy policies. Hippocratic Databases [4] have been proposed to implement such policies within a corporation's datastore. From an end-user individual's point of view, both of these rest on an uncomfortable philosophy of trusting corporations to protect his/her privacy. Recent history chronicles several episodes when such trust has been willingly or accidentally violated by corporations facing bankruptcy courts, civil subpoenas or lucrative mergers. We contend that data management solutions for information privacy must restore controls in the individual's hands. We suggest that enabling such control will require a radical re-think on modeling, release/acquisition, and management of personal data.

1 Introduction

Information Privacy is concerned with imposing limits on collection and handling of personal information such as credit and medical records by state and private organizations. These are early days for Information Privacy, and norms and laws that impose restrictions on the use of personal information collected by organizations are being worked out as a solution. Technology is being devised to assist the implementation of such laws.

Status The Platform for Privacy Preferences (P3P) [27, 32] is a set of standards that allow organizations to declare their privacy policies. Recently, Hippocratic Databases [4] were envisioned to provide support for an organization's privacy policies within the organization's datastore. Thus, in this framework, an organization would post its privacy policies, using agreed-upon language, and an individual would only conduct business with that organization if the published policies were consistent with the individual's expectations.

Example Consider an individual, Alice, who wants to sign up for a DealsRus service on the web. DealsRus requires Alice's email address to inform her of upcoming deals. DealsRus recognizes the privacy concerns of its clients and has placed its P3P policies on their web-site. Alice is privacy-savvy and is using a browser which is P3P enabled. Before giving her email address, Alice's browser would fetch the P3P policy from the DealsRus web-site. For instance, DealsRus may state that email addresses will only be used for current purpose ("completion and support of the recurring subscription activity") and the recipients of such data will be restricted to ours ("DealsRus and/or entities acting as their agents or entities for whom DealsRus are acting as an agent") but not unrelated third parties. If Alice is happy with this policy, then she can give DealsRus her email address.

Critique With the P3P framework, thus, Alice has to trust that (a) the organization has clearly stated its policies, that (b) the organization will actually adhere to the policies, and that (c) the organization has the means to implement the policies in transit and storage. All three aspects raise troubling issues: Even though DealsRus has used legal language vetted by P3P, the end user may feel inundated with legalese whose exact practicality is open to interpretation [21, 25, 30, 36]. For instance, what exactly does current purpose mean? Perhaps it is within the ambit of current purpose for DealsRus to spam their customer's mailboxes? And what does it mean not to give email addresses to third parties but restrict recipients to ours? Perhaps DealsRus has many wholly-owned subsidiaries which can use the addresses? Does DealsRus provide adequate protection for personal data to prevent easy access to data by intruders? And what would happen if DealsRus declares bankruptcy, or changes management, or changes it policies, or its records are subpoenaed by a court?

Inherency P3P and Hippocratic Databases put the onus of safeguarding privacy in the hands of organizations that are often themselves guilty of trespass or sloppiness. Indeed, recent history chronicles several episodes where corporations have violated, either deliberately or accidentally, their customer's trust when they faced mergers, bankruptcy, courts or hackers [2, 19, 23, 34]. Even if the underlying datastore at DealsRus follows Hippocratic principles, if the rules the datastore is told to follow by DealsRus management are not the "ethical" ones (as far as Alice is concerned), then the Hippocratic guarantees will be of little use to Alice.

Thesis We contend that there is a better way to approach privacy, and that is to enable individuals to retain "control" over their information. At all times, the individual should be able to "choose freely under what circumstances and to what extent they will expose themselves, their attitudes, and their behavior, to others" [41]. The desired "level of control" may vary: for instance, in some cases, the individual may only want that misuse of her information be auditable. In other cases, she may want to prevent access to information to certain organizations or for particular tasks. In any case, however, it should be the individual who pro-actively decides what the level of control is.

We believe that a case can also be made to organizations to leave control in the hands of individuals. In particular, governments are passing new legislation (e.g., California law SB1386, effective July 1, 2003) that forces organizations to inform individuals whenever there has been a privacy breach, and makes organizations liable for improper use of information. Given the high overhead of securing data, and potentially high liability costs, organizations could be persuaded to leave control to owners of the information.

Plan How can an individual retain control at the appropriate level once she has released information to an organization? Returning to our example, Alice would "retain control" if

A *Permission:* Every time DealsRus wanted to use Alice's email address, it would contact Alice to check if this particular use were approved;

B *No Copies:* Alice delivers her email address in a "magic" read-only fashion that makes it impossible for DealsRus to make copies;

C *Supervision:* Alice is allowed to supervise how DealsRus actually uses the one copy of her address, e.g., DealsRus will make its business processes transparent to Alice allowing her to see how her information is being used; and

D *No Integration:* Alice's email address value makes it impossible for DealsRus to acquire more information about Alice from third parties while using her email address as the integration (join) key.

Obviously, our control wish-list is a fantasy to say the least. Even if DealsRus would agree to ask Alice's permission each time her email address was going to be used, Alice may not want to be interrupted all the time. And even if it were feasible to audit an organization's business processes, DealsRus would never allow its proprietary processes to be made public. And so on and so forth.

In this paper, we illustrate through examples a series of scenarios where an individual can retain limited control over her information (Section 2). We claim that these examples are representative of a small set of "information types", and that for each such type, one can devise a general purpose set of mechanisms to retain control (Section 3). We then propose to gather the set of mechanisms that cover all information types in one framework which we call *P4P: Paranoid Platform for Privacy Preferences* (Section 4).

We call this framework "paranoid" because individuals that use it are less trusting of organizations than individuals who use the P3P framework. We caution that our framework is still in its formative stages, and many of our concepts are still not well-defined. Thus, at this point, we are only presenting the *vision* of our framework, with the hopes that others in the community will help us refine it, formalize it, and debug it.

2 Retaining Control – Initial Examples

We claim that it is possible to approximate the idealized control we have sketched using a variety of application-dependent techniques. Let us illustrate what we mean using a couple of examples: email addresses and credit card numbers. Note that all of the techniques we will exploit in these examples are well-known. As we will see after the examples, our goal will be to synthesize general mechanisms out of the well-known individual techniques.

2.1 Email Address

To retain control over her email address, Alice constructs a trusted software *agent* that manages her email address. (See Figure 1.) Only the agent is given Alice's true email address, say aly@aliceHost. When Alice wishes to give her email address to DealsRus, the agent generates a new address, say aly1@agentHost, where agentHost is the computer where the agent runs. When DealsRus receives aly1@agentHost (either from Alice or from the agent), it uses that address for communication with Alice. That is, an email to aly1@agentHost is received by the agent, and will be forwarded to aly@aliceHost depending on what restrictions Alice specified when the email was created.

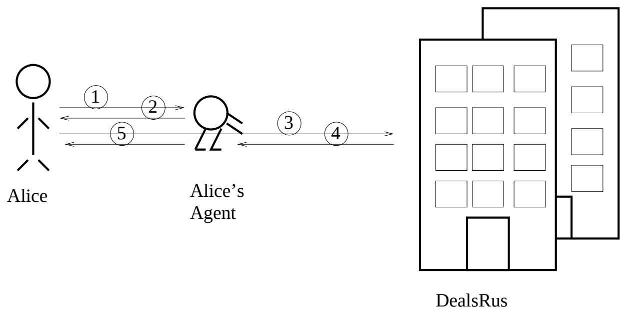

Figure 1: Alice interacts with DealsRus through her agent. Alice (1) requests and (2) obtains a new temporary email address from her agent, which is (3) released to DealsRus. Henceforth, DealsRus can only (4) send messages to the agent at the released temporary email address which are (5) forwarded to Alice after checking for specified restrictions.

Alice specifies restrictions to the agent to give her some control over how aly1@agentHost can be used. For example:

1 *Timeout:* The email aly1@agentHost is only valid for a period of time. After that time, the agent will refuse to forward messages to Alice.

2 *Limited Use:* The agent will only forward some maximum number of messages.

3 *Restricted Source:* The agent will only forward a message if it comes from a pre-specified source.

4 *Invalidation:* Alice can at any time explicitly instruct the agent to stop forwarding messages.

5 *Isolation:* The email aly1@agentHost is only released to DealsRus. Other organizations will get different email addresses.

How does Alice gain by trusting an agent? Alice now has to trust only *one* entity, as opposed to every organization that receives her address under the P3P framework. Furthermore, if the agent code is public, Alice can see precisely what actions the agent will take under different circumstances. This operational description of the agent's "privacy rules" can be much more precise that any legal/natural-language description that an organization provides under P3P.

The forwarding restrictions enforced by the agent do not provide all the control we wanted in our idealized wish list, but they do provide a very useful level of control for Alice. In particular:

A DealsRus can distribute copies of aly1@agentHost to other organizations, but a restricted-source limitation can prevent the other organizations from getting through to Alice. (Approximates No Copies goal mentioned earlier.)

B If DealsRus gives aly1@agentHost to other organizations, the agent will have proof of this action, since the address aly1@agentHost was only given to DealsRus. (Also approximates No Copies goal.)

C If the intended use of the address spans a limited time or a limited number of interactions (e.g., perhaps Alice is simply trying to find out what deals are available this week), then by implementing a timeout or a limited-use restriction Alice can ensure her address is not used for other tasks later in time or for tasks that require more interactions. (Approximates Supervision goal.)

D If DealsRus wishes to use Alice's address in some new way, it is likely that it will have to request permission from Alice (or her agent), as aly1@agentHost is likely to be invalid. (Approximates Permission goal.)

E If Alice uses different email addresses for each organization she interacts with, DealsRus will be unable to use Alice's address as an integration key to obtain more information about her from third-party organizations. (Approximates No Integration goal.)

As we can see, for the particular case of email addresses, Alice can retain a fair amount of control, and does not have to trust as much the organizations she deals with. Furthermore, organizations do not have to change their procedures; they handle email just as they did before.

Adoption and Challenges Notice that the agent can be implemented in a variety of ways. The agent could be part of Alice's desktop email software, or it could be a third party that provides the temporary email address generation and email forwarding service. Indeed, www.mailshell.com and www.spamgourmet.com provide such a third-party facility today with *Timeout* and *Limited Use* options respectively (also see the Lucent Personalized Web Assistant project [15]).

The temporary email addresses must be generated carefully. It should not be possible to infer the true address from the temporary one. Similarly, successive email addresses obtained by Alice should truly be independent from each other. Moreover, Alice should not be the only individual using the "@agentHost" addresses. How can we design an agent that resides on Alice's desktop to meet such constraints? Perhaps a P2P-based email service could be designed along the lines of [7, 35] in which Alice's agent can participate and hide in a "crowd" of other agents.

A trusted third party can hide Alice in its "crowd" of users. However, scaling with the number of users will be a challenge for such a provider. Recall that each user might wish to create a new email address for each organization that he/she deals with. Each such account might have a unique set of restrictions. Receiving, filtering and forwarding emails with low latency might be a challenge.

An interesting solution is for the third-party agent to encode the restrictions in the email address itself.

That is, the agent can encrypt the restrictions using its public key [20], and include them as part of the address. When email is received on an address, the agent can decipher the restrictions (using its private key) and enforce the rules. Such a scheme trades-off disk-accesses for runtime processing.

2.2 Credit Card Number

The above ideas can be extended to the handling of credit card numbers as well. An agent can ensure control of an individual, say Bob, over the use of his credit card. However, since credit is extended to Bob by his bank, we need to place the agent either at Bob's bank or between the bank and the organizations that use Bob's credit card number. If the agent is not at the bank, it would have to appear to the organizations as a bank that can handle charges, which may be difficult to achieve. Thus, let us assume that the bank plays the role of trusted agent for Bob.

The interaction between Bob and an organization, say ShopsRus, is analogous to that between Alice and DealsRus. Neither Bob nor his agent gives out Bob's true credit card number. Instead, ShopsRus receives a unique, temporary credit card number, which we will call a *pseudonum*. The agent manages the mapping between this pseudonum and Bob's credit card number. The agent may also enforces restrictions on the use of the pseudonum in a variety of ways:

1 *Timeout:* The time of validity or number of charges;

2 *Limited Use:* The total amount that may be charged to the pseudonum;

3 *Restricted Source:* The sites that may make charges, or the types of purchases that may be made to the pseudonum;

4 *Invalidation:* Bob can at any time explicitly instruct his bank to stop honoring charges on the pseudonum.

5 *Isolation:* A unique pseudonum is released to ShopsRus. Other organizations will get different pseudonums.

As with his email, Bob retains some level of control as to how his credit card is used.

A ShopsRus can distribute copies of the pseudonum to other organizations, but a restricted-source limitation prevents other organizations from charging Bob's credit (Approximate No Copies).

B If ShopsRus gives the pseudonum to other organizations, the agent has proof of this action, since the pseudonum was only given to ShopsRus (Approximate No Copies).

C If the intended use of the address spans a limited time or credit, then with an appropriate restriction Bob ensures that his credit is not used for other tasks later (Approximate Supervision).

D If ShopsRus wishes to charge Bob's credit for a new deal, it is likely that it will have to request permission from Bob (or his agent), as the original pseudonum is likely to be invalid (Approximate Permission).

E If Bob uses different pseudonums for each organization he interacts with, ShopsRus will be unable to use Bob's credit card number as an integration key to obtain more information about him from third-party organizations. (Approximates No Integration goal.)

Adoption Indeed, some credit card companies have begun offering a subset of the above functionalities (e.g., one-time use credit card numbers [6, 22]). The technology was hailed as a "landmark event by the industry" and promptly adopted by online merchants who have to bear the brunt of credit-card fraud, unlike offline merchants in which case the liability is assumed by the bank that issued the card. For example, the travel-site www.expedia.com recorded a fiscal third-quarter charge of six million US dollars in 2000 to cover the cost of fraudulent transactions! The above anecdotes suggest that organizations will indeed be willing to leave control in the hands of individuals if appropriate technology is devised.

3 Retaining Control – Generalizing to "Information Types"

Can we generalize these concepts, so that an individual can retain control over other "types" of information? It turns out that email addresses and credit cards are the easiest to control as they represent a "service handle" for a workflow path that terminates at the individual. An agent can easily place itself in this path and provide limited control on how the service is invoked. Indeed, this is why we already have deployments that seek to exercise control in ways described earlier.

In this section, we consider four types of personal information that are ubiquitous today: (a) Local Identifiers, (b) Foreign-Key Identifiers, (c) Value Predicates, and (d) Multi-Source Value Predicates. We will see that it will be harder to retain the same degree of control as with service handles, and organizations may have to dramatically change the way they handle personal information. Nevertheless, we are cautiously optimistic that a collection of techniques can be devised that may lead to the synthesis of a general framework.

3.1 Local Identifiers

In many cases, organizations demand from its users identification numbers like social security numbers (SSN) in the United States, or national identification numbers in other countries. For instance, the first thing that many mail-order or on-line stores ask for is a telephone number, since that is how they locate their customer's records. In many cases, these numbers are *only* used as keys in the local database, and nothing else.

Simple Protocol It is easy for an agent to hide the true identity of an individual, say Carol. The agent generates a unique, private identifier for Carol, which could be for example, a random 256 bit string. The organization, DealsRus receives this private identifier, and uses it as a primary key for Carol. The agent of course remembers all of Carol's identifiers, so whenever Carol needs to contact DealsRus, the proper identifier can be issued. And as in our previous examples, Carol retains control over her identity: DealsRus does not know Carol's true phone number or SSN, so any abuses of the private identifier are limited in scope.

DealsRus may only be willing to accept identifiers that look like SSNs or phone numbers. In such a case, Carol's agent must map the random identifier into one that conforms to what DealsRus expects.

Challenges There are a few practical issues that must be considered for such indirect identification numbers to work with today's deployed systems. There is a chance that some other individual will generate the same private identifier as Carol. Organizations may already have procedures in place for duplicate identifiers. For example, two family members who share a phone may be buying goods at the same mail-order store. Even SSNs are known not to be really unique identifiers, and conflicts do happen. The bottom line is that organizations need to be prepared to deal with duplicate user-generated identifiers, and should have a protocol in place to ask the user for a different one. Such protocols are already common at web-sites where users select their ID: if the ID is taken, the sites prompts the user for a different ID.

A potential scheme to ensure uniqueness is the following. Carol can provide her agent with a particular (unique and secret) data item, say her SSN. The agent will then generate all identifiers for Carol based on this data item and the organization's name, e.g., by using a one-way hash function like SHA-1 [14]. The identifiers that are generated can be shown to have good cryptographic properties: independence, uniqueness, and non-invertibility.

Real identifiers such as phone numbers have the advantage that they are more readily remembered by users. Thus, when Carol personally phones the DealsRus help line, it will be a lot easier for her to remember her phone number rather than the randomly generated identifier. Or perhaps Carol has allowed her phone to automatically provide her number to the callee (a feature known as caller-id in the United States), in which case DealsRus will immediately know who is calling. This example illustrates a classic privacy-convenience trade-off which cannot be avoided: If Carol wants privacy, then she is better off giving out agent-generated identifiers, and always relying on the agent for interactions with outside entities. If Carol wants convenience, then she can give out her personal identifiers, and hope that organizations do not do anything too bad with them.

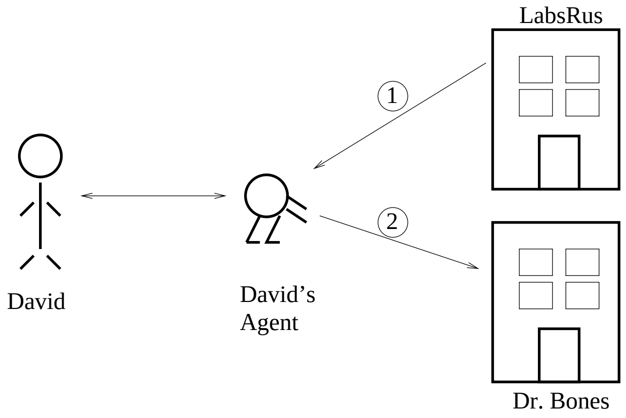

Figure 2: LabsRus interacts with Dr. Bones through David's agent. (1) LabsRus sends David's reports to his agent. (2) The reports are forwarded to Dr. Bones by the agent.

3.2 Foreign-Key Identifiers

In some cases, an individual's identifiers are used more than just as local identifiers. The organization may need to use them as foreign-keys to allow a legitimate (individual approved) integration or retrieval of records from other organizations.

To illustrate, say David is a patient at Dr. Bones' clinic, and had some tests done at LabsRus. (See Figure 2.) The patient information system that Dr. Bones uses identifies David by i_1, a local identifier generated by David's agent. Similarly, LabsRus identifies David by i_2, a different local identifier. When David gets a blood test at LabsRus, he requests that the results be sent to Dr. Bones' clinic. Thus, LabsRus needs a way to send records for i_2 that are received as records for i_1 at Dr. Bones' clinic.

Simple Protocols There are at least two ways to do this mapping:

A LabsRus can ask David's agent for David's identity with Dr. Bones. If the agent gives LabsRus i_1, then LabsRus can communicate directly with Dr. Bones. However, LabsRus now knows David's identity with Dr. Bones, and could misuse it later on. David would not know of any future sharing of information between LabsRus and Dr. Bones, and hence lose control of his personal information.

B LabsRus can route the blood test results to Dr. Bones through David's agent. One way to do this is as follows: David instructs his agent to anticipate blood test results from LabsRus that are to be routed to Dr. Bones. When LabsRus has the results, it sends a message to David's agent which includes:

1 David's local identifier i_2;

2 David's blood test results;

3 A signature that can be used to prove that only LabsRus could have generated the given test results;

At this point, David's agent removes the i_2 identifier and the signature of LabsRus from the message. The agent logs the signature as proof of the authenticity of the report, if needed later. The agent then adds the i_1 identifier and forwards the results to Dr. Bones.

Notice that in [B], LabsRus remains unaware of David's doctor who receives the reports. Dr. Bones is unaware of the place where the tests were performed. Thus, David again retains some control over his information. Anytime an organization wants to contact another organization to share David's information, it must go through David's agent.

Challenges In this scenario, it is clear that organizations will have to change the way they operate. That is, they need to be aware that following foreign keys needs to be done though agents, and not directly as they do today. We also observe that the above is still a sketch and a rigorous protocol needs to be defined that will allow such foreign-key mappings to be used via a trusted agent without leaking any personal information. For example, how can David's agent be assured that LabsRus is not hiding information within the test reports that reveals David's and its identity?

3.3 Value Predicates

In our next example, say Ellen is purchasing a cruise from ShipsRus, and the site asks her for her age. Let us assume that ShipsRus has a legitimate need for Ellen's age y. Perhaps ShipsRus offers a senior citizen discount to individuals with an age over 60 years who take the cruise. We can model this situation by saying that ShipsRus needs to compute a predicate $p := (y > 60)?\ \mathsf{true} : \mathsf{false}$. Thus, ShipsRus does not need to know y itself but the value $p(y)$. How can ShipsRus obtain $p(y)$ while Ellen retains control over her age attribute y?

Simple Protocol We can proceed as follows. ShipsRus sends the predicate p to Ellen's agent, e.g., as a SQL statement. (The query will run in a sandbox-ed environment so it cannot have undesired side effects.) Ellen gives y to her agent, which then computes $p(y)$ and sends the result to ShipsRus. In this way, Ellen retains control over her age information, and only gives ShipsRus the *minimal legitimate* information ($p(y)$) it needs to have to provide service to Ellen.

Challenges There are, of course, two important issues we need to consider. First, the organization can "cheat" by using predicates that are easy to invert. For example, ShipsRus may give the trusted agent a series of predicates that serve to identify Ellen's age y uniquely (e.g., $p_1 : (y == 58)?\ \mathsf{true} : \mathsf{false}$, $p_2 : (y == 59)?\ \mathsf{true} : \mathsf{false}$, etc.) The only way to avoid this problem is to have DealsRus disclose the nature of the predicates by making the source SQL code visible to scrutiny by Ellen and her agent. This way Ellen can understand the nature of the information that is being given to ShipsRus. For example, if p is the predicate given earlier that checks if age is greater than 60, Ellen will know that she is disclosing the fact that she is or is not a senior citizen to ShipsRus.

The second issue is that Ellen may cheat and not give her true age. Of course, cheating may have later repercussions for Ellen. For example, she may run into trouble when she shows up for the cruise with a senior discount ticket and looking like a teenager! We note that Ellen could cheat by giving a false age even if ShipsRus were to ask Ellen for her age directly.

Notary Protocol Is there anything that could be done to prevent cheating by Ellen? For instance, say ShipsRus does not trust Ellen, but does trust some other organization that can act as a *notary* and vouch for Ellen's age, like Dr. Bones office. Can we devise a protocol that enables Ellen to compute a ShipsRus predicate, whose result is vouched for by Dr. Bones?

We present a weak version of the notary protocol that requires Ellen to trust Dr. Bones not to divulge her information. With this protocol, the P3P guarantees are the best we can hope for. Let us say that Ellen discloses her age to Dr. Bones, e.g., by having a medical examination or by showing her birth certificate. There is no way Bones will vouch for Ellen's age without knowing the age, so we cannot avoid disclosing the information.

Ellen's agent must also disclose the mapping between the identifier Ellen used at DealsRus, i_1 and the identifier Ellen uses with Dr. Bones, i_2. Without the $i_1 : i_2$ mapping, Dr. Bones cannot really say whose age he is vouching for. Ellen's agent also provides a signature for the $i_1 : i_2$ mapping, so that if a dispute arises in the future, Dr. Bones can prove that Ellen's agent claimed that i_1 and i_2 were the same person. In summary, the request for Ellen's age proceeds as follows:

1 DealsRus asks Dr. Bones to evaluate $p(y)$ for the person DealsRus calls i_1 and who uses Ellen's agent.

2 Dr. Bones asks Ellen's agent for Ellen's id at DealsRus. If the agent approves, it sends the $i_1 : i_2$ mapping, appropriately signed.

3 Bones then looks up i_2's age, y and computes $p(y)$. The result is returned to DealsRus.

Challenges The weaker protocol we presented requires Ellen to reveal her DealsRus id to Dr. Bones. The notarizing organization (Dr. Bones) could thus gradually get to know Ellen's identity at various organizations.

Of concern to DealsRus is the fact that it has to reveal its predicate p to Dr. Bones. Is it possible to devise a protocol that allows DealsRus to compute $p(y)$ when DealsRus does not know y and Dr. Bones does not know p?

Trusted Third Party In the above protocol, Ellen's agent could cheat and provide a false mapping since only it knows the relationship between Ellen's two personas. An improvement would be to run agents at sites that could be trusted by both Ellen and the organizations she deals with. In such a scenario, a trusted organization, which we can call the *agency* can run privacy agents for a variety of individuals. The agency somehow gathers evidence that a particular individual is who they say they are (perhaps they have to show up in person and identify themselves with a photo identification), and then runs an agent on their behalf. The code used for the agent can be public so that customers gain trust in the provided services. Even if the agency is trusted, individuals can still cheat in various ways, but at least organizations are able to go to the agency for help in resolving conflicts that may arise.

As far as Ellen is concerned, her agent is a part of the agency. Thus the mappings of its personas are known only to the agency, and need not be revealed to Dr. Bones. DealsRus still has to reveal its predicate p to the agency. All parties now have to trust only *one* organization and need to rely on its P3P promises.

3.4 Multi-Source Value Predicates

The need for parties trusted by individuals and organizations becomes more evident if we consider more complex scenarios where predicates need values from different sources. To illustrate, consider a bank, EasyLoan, that needs Fred's age and salary in order to determine if it can give him a loan. All that EasyLoan needs is the output of a predicate $p(y, s)$, where y is Fred's age, and s is his salary. Fred does not want to disclose either attribute to EasyLoan, and EasyLoan does not trust Fred to compute $p(y, s)$. Fortunately, EasyLoan does trust Fred's employer, Acme, to provide the salary, and Fred's doctor, Dr. Bones, to provide the age needed by the computation. How can EasyLoan obtain $p(y, s)$ from values provided by Acme and Dr. Bones while Ellen retains control over her age and salary attributes?

Secure Multi-Party Computation Cryptographers have studied ways to evaluate arbitrary functions in a distributed fashion. The goal of secure multi-party computation is to compute a function $f(a, b, c)$ with inputs a, b and c at three different parties, such that the three parties learn only $f(a, b, c)$ and nothing else. Yao [42] showed that any multi-party computation (and hence our multi-source value predicates) can be solved by building a combinatorial circuit, and simulating that circuit. However, such schemes incur excessive computation and communication overhead. Can the recent work [18, 33] in this area makes evaluating specific predicate queries more practical?

Trusted Third Party A simpler solution is to use an agency that is trusted by all parties. The steps to do the computation are then as follows: EasyLoan asks the agency to compute $p(y, s)$ for the person it knows as i_1 and who uses a particular agent. By this point, Fred has already disclosed to EasyLoan who may provide the age and salary, so the request from EasyLoan also includes the identities of Acme and Dr. Bones. The agency then asks Fred's agent for Fred's identities at Acme and Bones, and asks these organizations for the required data. Finally, the agency computes $p(y, s)$ and returns the value to EasyLoan. The agency keeps a record of the computation in case of future disputes.

4 P4P: Paranoid Platform for Privacy Preferences

We have illustrated through examples a set of information types where an individual can retain control over his information. We claim that for each such information type, one can devise a general-purpose set of mechanisms to retain control. We propose to gather these set of mechanisms into one framework, which we call *P4P: Paranoid Platform for Privacy Preferences.*

We believe that private information can be classified along three dimensions: (a) ownership, (b) function, and (c) desired level of control. In this section, we illustrate the above properties and glean principles that can underly the P4P framework. We caution that our framework is still in its formative stages, and many of our concepts are still not well defined.

For our illustration, we need to refer to a data model, and here we chose a simple entity-attributes model, although of course other models are possible.

- *Attribute:* As usual, attributes represent the basic building blocks of information, and let us say they are (label, value) pairs. For example, (name:"Alice") and (address:"123 Main St.") are attributes. To simplify our notation, we will remove the quotation marks from string values.

- *Entity:* An entity has a set of related attributes. We are especially interested in entities that represent individuals or organizations. For ease of notation, we will use the term "entity" to refer to both (a) an individual/organization P in the real world, and (b) the representation of P in our framework as a set of related attributes. For instance, Alice is an entity that may be represented

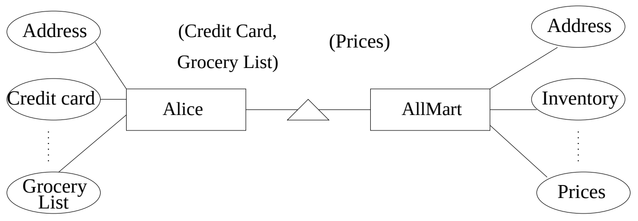

Figure 3: Rectangles represent entities, Ovals are attributes of entities and Triangles are interactions between entities.

by the following set of attributes: [Name:Alice, Address: 123 Main St., Phone: 555-1234].

Note that in our world, attributes by themselves are usually not sensitive, e.g., nobody will care if someone knows the attribute phone: 555-1234. It is only the *association* of phone: 555-1234 and name: Alice with the entity Alice that is sensitive information. Thus, typically associations of attributes with an entity may be considered sensitive information.

Each entity has a datastore whose contents can be classified by the data-item's ownership, function and desired level of control. Each entity also associates a trust level for other entities it deals with, and uses those trust levels to determine how to interact with them. Each interaction involves exchange of data between the participant entities that reveals attributes of one entity to others.

Example [*Entity Interaction*] Suppose Alice buys *bread* and *butter* at AllMart using her credit card. The interaction between Alice and AllMart as seen in our framework is shown in Figure 3. Alice is an entity with attributes address, credit card information and a grocery list. Similarly, AllMart is an entity with attributes address, inventory and prices of goods. Alice's interaction with AllMart reveals information about one entity to the other. Thus, Alice must share her credit card information and grocery list with AllMart. AllMart must reveal its prices of goods to Alice.

Ownership Consider the interaction between two entities, an individual Alice and an organization AllMart. The participant entities share data with each other. The data that is shared can be owned by Alice or by AllMart.

- *By Individual:* This category includes data generated by Alice, i.e., her personal data (e.g., address) and preferences (e.g., grocery list). The fact that Alice owns this data means that Alice should retain full control over it, deciding when and how it can be revealed to others.
- *By Organization:* This category covers data generated by AllMart. For instance, say Alice buys a particular book there. The fact that someone purchased a book may be represented by a tuple <invoiceNo:123, bookTitle: War and Peace>. This tuple is owned by AllMart, not Alice. Even though it was a purchase by Alice, Alice cannot control the information. However, Alice should own the information that links this purchase to her, e.g., the tuple: <invoiceNo: 123, name: Alice, address: 123 Main St.>.

Function In order to control information, one needs to know what role it plays vis-a-vis interactions between entities. This role can be of the following types, as illustrated by our examples in Section 3. Note that the same information can be of multiple types, depending on how it is used in an interaction. For instance, an email address could be a service handle, and it could also be an identifier.

- *Identifier:* An identifier is an attribute (or a set of attributes) that is used to identify an entity in a datastore. For example, if Carol gives her phone number to DealsRus, then DealsRus may use that number to refer to Carol in its interactions with Carol (or her agent).
- *Service Handle:* A handle is an attribute that provides a path to a service that some other entity provides on behalf of Alice. Email addresses and credit card numbers are examples of service handles.
- *Input to Predicate:* An attribute is of this type if it can be used as an input to predicates other entities wish to evaluate. In our examples, age and salary were attributes of this type.
- *Copy:* Attributes can be copied to another entity's datastore. In such cases, it is critical to track which of the copies is the *primary* copy (at the site that owns the information) and which are the *secondary* copies.

Desired Level of Control This level specifies how the owner wants the information managed. The level may vary by entity, that is, one level may be desired for one entity, and a different level for another, probably depending on the trust level placed on the entities (see below). Also, the desired level can refer to a set of attributes, since as we have discussed, a subset may be more sensitive than others. Based on our examples in Section 3, we can list the following useful points in the spectrum:

- *Complete Privacy:* The information should not be revealed at all.
- *Limited Time/Use:* The information can be used for a limited time, or a limited number of times, or for a particular task, as in our email example.

- *No Predicate Input:* The information cannot be used as input to predicates. We may disallow either single-source or multi-source value predicates.

- *No Integration:* The information is given to an entity, but that entity should not be able to integrate that information to other subsets we have given other entities. To enforce this restriction, we need to control the foreign keys we give out.

- *Accountable:* The information is given to an entity, but that entity should be held responsible for any misuse of the information in the future.

- *Sharable:* As we have argued, there are cases where we must give information to an entity and hope the entity will protect our data. For this type of sharable data, we may specify what guarantees we may want from the entity, as in the P3P framework. For instance, we may only share information with an entity if it promises not to divulge it to third parties, or if it promises to only use it for computing statistics.

Trust As mentioned earlier, in our framework, an entity also needs to specify the trust it has in other entities. There are many ways to specify and manage trust that have been discussed in literature [1, 9, 16]. Given our focus on controlling access to information, one option is to simply specify policies stating the level of control desired on information released to a target entity. For example, if an entity is trusted to enforce no-integration, then we believe it will not attempt to integrate the information we give it with what is available at other entities. If we trust an entity in this fashion, then we do not have to implement precautions with the identifiers we give it.

Properties of Interaction An unconstrained exchange of information in interactions can reveal an entity's private attribute values to others. We propose that information that is exchanged during an interaction be carefully "trimmed" to ensure information privacy. For example, let Alice participate in successive interactions $I_1, I_2, I_3, \ldots$ with DealsRus. To ensure Alice's privacy, we require the following, which we call the *TRIM* properties, on each interaction:

- *Traceability*: The data that is exchanged during an interaction I_j cannot be used by DealsRus for an interaction with another entity, without Alice having proof of DealsRus' involvement.

- *Revocability*: Alice can "severe" associations with a particular interaction in the future (e.g., on expiry of a subscription); the attribute values that was shared in such an interaction cannot be associated with Alice anymore.

I. Attribute	*II. Function*	*III. Level of Control*
Image	Copy	Accountable
Email	Service Handle	Limited Use
Email	Identifier	No Integration
Age	Input to Predicate	No Input

Table 1: *P4P policies defined by Alice for interactions with DatesRus.*

- *Isolation*: Data provided in two interactions I_j and I_k $(j \neq k)$ cannot be associated with the same entity Alice.

- *Minimality*: Alice ensures that the data that is exchanged in an interaction is the minimal to successfully achieve its goals.

To illustrate our proposed taxonomy, let us return to our sample entity for Alice [Name:Alice, Age: 23, Email: alice@public.org, Image: myPic.jpg]. Let us assume that Alice wants to sign up with an online dating service called DatesRus, but does not trust DatesRus with her personal information. Alice may define P4P policies as shown in Table 1 to govern her interactions with DatesRus. Each row in the table illustrates a policy that enables Alice to retain control over her personal information vis-a-vis DatesRus. Column I lists examples of attributes over which Alice desires control, Column II specifies the attribute's function, for which Column III specifies the level of control desired. We consider each row in more detail next.

Example [*Image* $\mapsto$ *Copy* $\mapsto$ *Accountable*] Alice wants to upload her image at the DatesRus site. However, she is worried about abuses: e.g., DatesRus may sell her image to advertisers without seeking her permission. Since she needs to make a copy of the image which will be shared with DatesRus, she deems her attribute image: myPic.jpg to have function: copy and level of control: accountable. Therefore, Alice's agent must ensure that the image provided satisfies property: traceable so that Alice will (eventually) obtain proof of DatesRus' identity. Alice's agent may watermark [24] the image to ensure traceability.

Example [*Email* $\mapsto$ *Service Handle* $\mapsto$ *Limited Use*] Alice wants to provide an email address for DatesRus to inform her of possible interests. However, Alice does not foresees herself using DatesRus for more than an year. She does not want DatesRus to contact her once she ends her membership. So she deems her attribute email: alice@public.org to have function: service handle and level of control: limited use. Therefore, Alice's agent must ensure that the released email address satisfies property: revocable. Alice's agent may provide a temporary email address that can be invalidated by Alice at will.

Example [*Email* $\mapsto$ *Identifier* $\mapsto$ *No Integration*] Alice wants to interact with various organization (e.g., DatesRus, DealsRus, ShipsRus) each of which wants her email address. Alice realizes that her email address is unique to her, and could be used as her identifier by the organizations. She does not want (a subset of) these organizations to get together and integrate their datasets without her knowledge. So she deems her attribute email: alice@public.org to have function: identifier and level of control: no integration. Therefore, Alice's agent must ensure that the released email address satisfies property: isolation as well. Alice's agent may create distinct temporary email addresses, one each for each organization to ensure isolation.

Example [*Age* $\mapsto$ *Input to Predicate* $\mapsto$ *No Input*] DatesRus has promotional offers from local clubs that provides free entry to DatesRus clients under the age of 25. Alice does not want to reveal her age and has decided to decline any offer that requires her to reveal her age. So she deems her attribute age: 23 to have function: input to predicate and level of control: no input. Therefore, Alice's agent must ensure that optional age-based predicates from DatesRus are not evaluated.

In summary, in our P4P framework, it is important to understand who controls (owns) data, how the data is being used (function), what control is desired, and what agents can be trusted. For each type of information (a point in the function, ownership, control space), our goal is then to provide one or more mechanisms that enforce the desired property.

In the P4P framework, trusted agencies play a central role. As illustrated in our examples, they provide agent and predicate evaluator services, so that entities can effectively control and at the same time share their information. Each individual would contract with one or more agencies to provide services, and perhaps to store some of their data too. As the individual interacts with organizations, instead of giving out information directly, it asks its agent to provide appropriate attributes, whether they be private email addresses or private identifiers. When an organization needs additional information about an individual, it can contact its agent or a trusted agency to obtain the data.

5 Challenges

We have sketched our vision for an information processing world where individuals can retain, whenever possible, control over their information. Organizations can also benefit by not getting control, and the liability that goes with it, of information they do not own. This information processing model will require that organizations and individuals operate with information in different ways (e.g., through agents, use TRIM interactions, etc.). Of course, the challenges to achieve this vision are huge, and in closing we mention a few.

5.1 Interfaces for Entities and Agents

Adequate programmatic interfaces need to be defined for entities, agents, agencies, predicate evaluators and notaries. Agent interfaces for dealing with information types will have generic and application dependent parts. For example, an agent may be asked to create a service handle that is limited for one day (a generic restriction) or a handle that only allows charges of up to 100 dollars (application specific for money-related handles). Traceable copies of data may require embedding of application-dependent fingerprints [24]. It will be important to explore the types of application-specific controls and services that would be useful.

5.2 Human Interfaces

Human interfaces must be invented that enable people to describe their privacy goals and select appropriate policies for their agents. The interface must also educate people about risks of their options. Can the recent work on privacy interfaces for ubiquitous computing be useful here? Research there has highlighted that individuals tend to release information subjectively while weighing in factors like information function, information sensitivity and trust in recipient [3, 29] which mirror our owner - function - level of control dimensions. Perhaps user-interfaces defined in the above context (e.g., SecureID [11]) can be adapted here.

We have already noted in Section 3.1 the trade-offs between convenience and privacy. Tools must be built that can integrate agents seamlessly with an individual's day-to-day tasks. For example, a web-browser toolbar could be built and helps the user obtain temporary email addresses from his/her agent.

There has recently been an interest in exploring the nature of privacy as a value determined by market forces [12, 26, 28, 40]. Instead of a declarative policy, individuals in this model may be willing to relax their level of control in return for a fair compensation. How can such schemes be incorporated in the interface, and indeed, the framework?

5.3 Reasoning about Information Privacy

While we have presented a few useful points in the ownership - function - level of control spectrum, it is important to specify information workflows for a variety of interactions and formally reason about privacy guarantees as an aggregate of an entity's interactions.

Suppose that an entity could log all interactions it participated in with other entities. How would such logs augment agent services? The agent can now use an entity's log to pre-process (or even abort) current interactions to prevent violation of the entity's privacy policies. For example, Alice's agent may raise an alarm if a sequence of predicates received in successive interactions with DealsRus are of the form $y == A?$ true :

false for $A = 58, 59, 60, \ldots$. Perhaps research in statistical databases [13] can be used to analyze predicates and detect such privacy breaches.

The above scheme works if we assume a "closed-world" [39] model, where an entity has only its own interaction logs to profile other entities. What would an "open-world" model entail with its assumptions of auxiliary datasets that may not currently be known to the individual's agent?

5.4 Architecture of a Privacy Agent/Agency

We touched upon challenges in designing privacy preserving protocols in Section 3. Perhaps the recent advances [10] in designing efficient group signatures [37] for anonymous authentication can be used to devise a Notary Protocol? A group signature scheme allows a member M of a group G to sign messages on behalf of G such that the resulting signature does not reveal M's identity. Thus, in the examples of Section 3, we could place Acme in a group of employers and Dr. Bones in a group of doctors. Acme and Dr. Bones would vouch for Fred's salary and age using their respective group signatures. EasyLoan can verify the signatures, and still not know the identity of Acme or Dr. Bones.

The examples in Section 3 assumed a cryptographic definition of privacy. Can efficient agents be designed for other relaxed notions of privacy (e.g., anonymity)? In an anonymity-based architecture, the agent should enable an entity to increase the level of anonymity of interactional data by using various information hiding schemes (e.g., k-anonymity [38], perturbation [5]). The infrastructure should, however, provide statistics to indicate the level of anonymity achieved. How can such statistics be maintained?

5.5 Trust Management

It will be important to understand the interactions between the P3P privacy policies and our privacy control mechanisms. The P3P framework still plays an important role, for describing how trusted organizations will manage data they own, or data they must absolutely have a copy of. Perhaps the agency can play a role in managing trust for the entities it represents. For example, it can track privacy breaches (e.g., misuse of limited-use emails or pseudonums) by organizations and assign them "trust ratings" (a la TRUSTe seals at www.truste.org). Such trust ratings can be used by individuals to determine appropriate policies for their interactions with an organization.

6 Final Thoughts

We contend that technology must be devised to allow individuals to retain control over their information. While other commentators have also stressed the need and suggested legal avenues [8, 17, 31], we have sought to devise an information processing framework that would enable such control.

For instance, cryptologists have provided primitives (e.g., Public Key Infrastructure [20]) to individuals to achieve communication privacy. We remark that while communication privacy could have been ensured by legislating that organizations (e.g., email servers, ISPs, employers, etc.) respect individual privacy, PKI primitives have put the right to communication privacy firmly into the individual's hands. Can we, in the database community, build an analogous infrastructure that ensures information privacy?

References

[1] A. Abdul-Rahman and S. Hailes. A distributed trust model. In *Proc. of the New Security Paradigms Workshop*, 1997.

[2] Acxiom opts out of opt-out. http://www.wired.com, 11/17/2003.

[3] A. Adams and M. A. Sasse. Taming the wolf in sheep's clothing: Privacy in multimedia communications. In *Proc. of ACM Multimedia*, 1999.

[4] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In *Proc. of the Conference on Very Large Databases (VLDB)*, 2002.

[5] R. Agrawal and R. Srikant. Privacy preserving data mining. In *Proc. of the ACM Conference on Management of Data (SIGMOD)*, 2000.

[6] American express offers disposable credit card numbers for online shopping. http://www.computerworld.com, 9/7/2000.

[7] M. Bawa, R. J. Bayardo Jr., and R. Agrawal. Privacy-preserving indexing of documents on the network. In *Proc. of the Conference on Very Large Databases (VLDB)*, 2003.

[8] J. Berman and D. Mulligan. Privacy in the digital age: Work in progress. *Nova Law Review*, 23(2), 1999.

[9] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proc. of the IEEE Symposium on Security and Privacy*, 1996.

[10] D. Boneh. A new life for group signatures. Plenary Talk at RSA Conference, 2004.

[11] D. Boyd. Faceted id/entity: Managing representation in a digital world. *MIT Media Lab, Master Thesis*, 2002.

[12] G. Calzolari and A. Pavan. On the optimality of privacy in sequential contracting. *Manuscript*, 2003.

[13] D. Dobkin, A. Jones, and R. Lipton. Secure databases: Protection against user influence. *ACM Transactions on Database Systems*, 4(1), 1979.

[14] D. Eastlake and P. Jones. Us secure hash algorithm 1 (sha1). *Network Working Group*, RFC 3174, 2001.

[15] E. Gabber, P. Gibbons, Y. Matias, and A. Mayer. How to make personalized web browsing simple, secure and anonymous. In *Proc. of the Conference on Financial Cryptography*, 1997.

[16] Y. Gil and V. Ratnakar. Trusting information sources one citizen at a time. In *Proc. of the International Semantic Web Conference*, 2002.

[17] J. Goldman. Personal information and privacy. In *Computers, Freedom and Privacy*, 1991.

[18] O. Goldreich. Secure multi-party computation. Working Draft, 2000.

[19] Hacker accessed customer information, acxiom reports (08/07/2003). http://www.siliconvalley.com.

[20] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. *Network Working Group, RFC 3280*, 2002.

[21] C. D. Hunter. Recoding the architecture of cyberspace privacy: Why self-regulation and technology are not enough. http://www.asc.upenn.edu/usr/chunter/ p3p.html, 2000.

[22] iPrivacy's Internet Identity Protection Service. http://www.iPrivacy.com.

[23] Jetblue shared passenger data. http://www.wired.com, 09/18/2003.

[24] S. Katzenbeisser and F. A. Petitcolas (Editors). *Information Hiding Techniques for Steganography and Digital Watermarking.* Artech House, 2000.

[25] J. Kaufman, S. Edlund, D. Ford, and C. Powers. The social contract core. In *Proc. of the International World Wide Web Conference*, 2002.

[26] J. Kleinberg, C. H. Papadimitriou, and P. Raghavan. On the value of private information. In *Proc. of the Conference on Theoretical Aspects of Rationality and Knowledge*, 2001.

[27] M. Langheinrich(editor). A P3P preference exchange language 1.0 (appel1.0). *W3C Working Draft*, 2001.

[28] K. C. Laudon. Markets and privacy. *Communication of the ACM*, 39(9), 1996.

[29] S. Lederer, J. Mankoff, and A. K. Dey. Towards a deconstruction of the privacy space. In *Proc. Workshop on Ubicomp Communities: Privacy as Boundary Negotiation*, 2003.

[30] K. Lee and G. Speyer. Platform for privacy preferences project (P3P) and Citibank. http://www.w3.org/P3P/Lee_Speyer.html, 1998.

[31] Lawrence Lessig. Architecture of privacy. cyber.law.harvard.edu/works/lessig/ architecture_priv.pdf, 1998.

[32] M. Marchiori(editor). The platform for privacy preferences 1.0 (P3P1.0) specification. *W3C Proposed Recommendation*, 2002.

[33] M. Naor and K. Nissim. Communication preserving protocols for secure function evaluation. In *Proc. of the ACM Symposium on Theory of Computing (STOC)*, 2001.

[34] Northwest airlines faces privacy suits. http://www.washingtonpost.com, 01/22/2004.

[35] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[36] M. Rotenberg. What larry does'nt get: Fair information practices and the architecture of privacy. *Stanford Technology Law Review*, 1, 2001.

[37] D. Schaum and E. van Heyst. Group signatures. In *Proc. of the EuroCrypt Conference*, 1991.

[38] L. Sweeney. k-anonymity: A model for protecting privacy. *Intl. Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5), 2002.

[39] J. D. Ullman. *Principles of Database and Knowledge-Base Systems, Volume 2.* W H Freeman, 1990.

[40] H. R. Varian. Economic aspects of personal privacy. *Privacy and Self-Regulation in the Information Age*, NTIA, 1997.

[41] A. Westin. *Privacy and Freedom.* Atheneum, 1967.

[42] A. C. Yao. How to generate and exchange secrets. In *Proc. of the ACM Symposium on Foundations of Computer Science (FOCS)*, 1986.