Simplified Privacy Controls for Aggregated Services — Suspend and Resume of Personal Data

Matthias Schunter, Michael Waidner

IBM Research, Zurich Research Laboratory
Säumerstrasse 4, CH-8803 Rüschlikon, Switzerland
{mts,wmi}@zurich.ibm.com

Abstract. The Internet is moving towards dynamic and ad-hoc service composition. The resulting so-called Web 2.0 sites maintain a unified user-experience while interacting and exchanging personal data with multiple other sites. Since the interaction is dynamic and ad-hoc, existing privacy policy mechanisms are not designed for this scenario.
In this article we describe a new lightweight approach towards privacy management. The core idea is to provide a "privacy panel" – a unified and simple entry point at each site that enables consumers to review stored data and manage their privacy. Key aspects were ease-of-use and handling of recursive disclosures of personal data.

1 Introduction

Increased exposure of web services by enterprises has lead to an emerging service aggregation ecosystem. Today, there exists no easily usable concept for distributed privacy controls in such federated service aggregations. As a consequence, today's service aggregations are either limited to applications in which no personal information is handled or to individuals that do not care about their privacy.

In this article, we describe a new concept of privacy panels for end users. We define a powerful yet usable mechanism that enable individuals to control their privacy throughout a network of aggregated services. Our main objectives are

- **Transparency**: Individuals can discover the privacy policy of a site, which data was collected, how it was used, and to whom it was disclosed.
- **Control**: Individuals can control data that is stored about them. This includes deletion or blocking and unblocking of data across all or some of the services that have been aggregated.
- **Usability**: Existing privacy enforcement concepts [5, 9] are powerful in an enterprise setting. However, they lack the ease of use and simplicity needed in an end-user-oriented scenario.

The privacy panel (see Figure 1 for an example picture) allows an individual consumer to manage privacy of a given site and all sites to which personal data has been disclosed by that site. The panel provides a single entry point to review the policy ("our policy") and the stored data ("your data"), to block and unblock further usage of portions of personal data ("block identity"), and to delete personal data ("delete identity").

The goal of our concept is to enable enterprises to act as better guardians of their customers' data.[1] Today, enterprises are often limited by the complexity of privacy concepts. As a consequence consumers suffered from limited transparency and control. Note that our concept needs to be augmented by proper auditing and controls to ensure that enterprises correctly deploy the technology and comply with the privacy promises they have made.

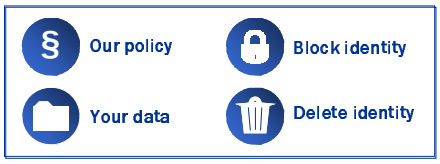

Fig. 1. Privacy panel

Outline The remainder of the article is structured as follows: Section 2 outlines the basic model of usable privacy control across multiple organizations and provides more details on the proposed privacy panel. Section 3 formalizes privacy controls in a single organization. Section 4 defines our trust management model and expands these concepts to protect personally identifiable information that has been disclosed to other organizations. Section 5 describes how to provide an enhanced level of verifiability to end users. Section 5.4 concludes our article.

2 Usable Privacy Controls in Aggregated Services

Our approach has three main components. The simple user interface to provide transparency and control to end users, the protocols that define how to implement the corresponding privacy controls across multiple organizations, and the policies and their semantics that allow organizations to formalize how data may be used. We use an online retail scenario to illustrate our concepts.

2.1 Online Retail Scenario

Consider the following scenario, involving a typical online bookstore B and a customer C (see Fig. 2). Customer C has an account with bookstore B, and B links whatever it knows about C to his or her account: all purchases made by C, voluntarily provided preferences, which books C looked at in the last few weeks. All in all, B has a fairly complete picture of C, as far as C's reading interests are concerned. By cross-correlating this data B gets a good idea about which books C might purchase in the

[1] Note that this approach augments privacy by means of self-protection [6, 7]: While self-protection minimizes the data that is being released and traces accidental disclosures, we focus on data that needs to be released despite minimization.

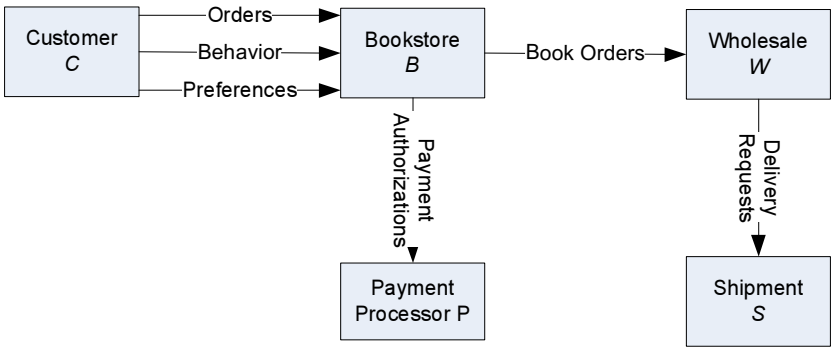

Fig. 2. Online retail scenario

future, which allows B to offer very precise recommendations to C. Unlike blind mass-marketing, these recommendations are precisely to the point, and thus C is likely to appreciate those recommendations and love this service.

Now assume that C buys a specific book at B, and pays with a credit card issued by a payment processor P. In order to fulfill the order, B sends certain data to P, for payment authorization, and then sends the book and shipping details to a wholesaler W, and finally this wholesaler sends the book plus shipping details to a shipping agent S. At the end, C's data went from B to P and W and from W to S.

Such scenarios raise significant privacy concerns: many people appreciate receiving specific recommendations, but they do not like the idea that others might see their behavior. Particularly critical situations are subcontracting scenarios (B delegates shipping to W, who further delegates to S), and acquisition scenarios. In the latter the fear is that when B' acquires B, and with it all of B's customer data, the business model of B might change, and suddenly the new business owner might decide that the new model will benefit more from selling the customer data to whoever pays most than from keeping them confidential.

2.2 Transparency and Control using a Privacy Panel

By adding a standardized privacy panel to all participating sites, a single entry point allows individuals to exercise control over their data. The panel is linked from all places where an enterprise collects or displays personal data (e.g., the login page, the page where C can inspect all previous orders), and will also create a specific identity management "portal" at a well known address. Say, if B can be reached at `http://www.B.com` then this portal will be at `http://www.B.com/identity`. The four icons in Figure 1 represent buttons on a web page. Ideally, these buttons and their essential semantics will be standardized so that customers who see them spontaneously and dependably associate the right meaning with them.

"Our policy" will open a window with C's privacy policy (most web sites offer this already).

"Your data" will open a window where C gets a report of all data B stores related to C. A standard should decide what "all data" will mean, but intuitively this is the data itself, plus the history of data (when and from whom and why did B receive data, and to whom and when and why did B send data?), plus links to the privacy panels of all parties that received data from B. If applicable, the panel should also explain how data was collected or from whom it has been received. Note that this includes direct and any indirect user data an organization plans to collect, i.e., if the collection of indirect data such as clickstream data is not declared, the organization is not authorized to collect this type of data.

"Block identity" will prevent B from using C's data for almost all purposes. All exceptions must be pre-agreed in the policy. Intuitively, these exceptions will only be purposes either required by law or needed to allow C to execute an *"unblock"* command. We call them "mandatory purposes", and if a data element is needed for at least one mandatory purpose we call it "protected data". The set of mandatory purposes is likely to be time-dependent. E.g., a transaction might end such that protected data becomes optional over time. In this case an earlier block (or delete) might have a delayed impact on such data. In our initial scenario we assumed an all-or-nothing scope for "block" and "unblock". A real implementation might give the user some choices. For instance, all data might be sorted into a few categories, and block/unblock and delete might be individually offered for each category. In the most extreme case, each data item can be blocked, unblocked and deleted independently of all other data items. Our technical description covers all cases, but we believe that few all-or-nothing choices for data groups are probably the most usable and thus most relevant case.

"Delete identity" is like "block identity" except that the effect cannot be reversed.

2.3 Related Technologies

Our proposal is related to privacy controls for individuals as well as privacy management based on privacy policy languages. Privacy policies fall into three main categories (see Fig. 3): Privacy notice from enterprise to consumers [12], privacy preferences of individuals [1], and policies governing data handling inside an enterprise [5, 9], The policy formalism used for explaining our mechanisms focuses on privacy notice and only provides high-level constraints for the enterprise internal use. The core goal of the formalism given is to provide an easy-to-understand formalism to describe the data flows between web-sites as well as a high-level summary of their site-internal use. Our simplified approach to policies can be augmented by detailed data types [12] and by mechanisms to validate whether the policies actually enforced satisfy the published promises [11, 8, 2].

Besides privacy policies, many existing concepts that enhance end-user control relate to our approach. In the sequel, we discuss some of them.

Unsubscribe. Many subscription-based information services, like electronic newsletters, allow customers to unsubscribe explicitly. The meaning is obvious: unsubscribe terminates the service for this customer, and in many cases the basic customer record

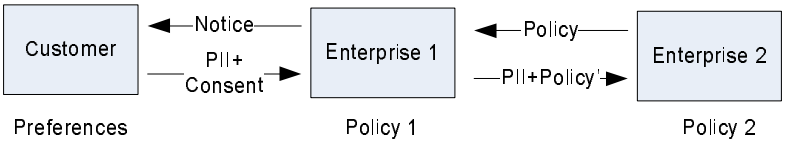

Fig. 3. Privacy policies and negotiation types

will be automatically deleted after some time. Unlike *block*, unsubscribe has no reverse operation (subscribing again does not recreate the old customer record such as an interest profile, but generates a new one). It has no transitive semantics (unsubscribe has no impact on other service providers who might have received the subscription information – this is actually a very common scenario with free subscription services), moreover, it also has no meaning for data beyond the basic customer record (e.g., if the service is a forum or newsgroup then all postings will still be available to all other subscribers).

Opt-in and opt-out choices . Organizations are supposed to specify the purposes for which they collect personal data. A popular way to specify those purposes is to structure them into a multiple-choice menu. For each menu item the user can say "yes", i.e., opt-in, or "no", i.e., opt-out. Often users can modify their choices at any time. Opting-out from a purpose is very similar to blocking, and subsequently opting-in again is very similar to unblocking. But there are differences:

- Opting in/out has no impact on other organizations. This is the same situation as with *unsubscribe.* One might argue that many proposals for privacy authorization systems suggest that policy changes must propagate ("sticky policies"), and opt-in/opt-out choices are elements of a privacy policy and thus should be sticky, too.
- Opting in/out impacts purposes only. For instance opting out from receiving a newsletter does not make the address inaccessible nor does it touch the history of how the customer reacted to previous newsletters. It just withdraws the organization's right to use the address for sending out a newsletter. Our approach is more general, in that it allows opt-in/opt-out for arbitrary elements of a concrete access control list.
- We also suggest specific implementations for opt-in/opt-out that have not been suggested in this context before. We suggest to off load storing or protecting data needed to unblock or opt-in from the organization to the user.

Several other technologies, such as content management and databases, are related in the sense that they provide functions that would make it easier to implement our concepts.

3 A Simple Policy Model for Local Privacy Enforcement

We now formalize the semantics of the proposed privacy panel. We start with a very basic definition. Consider an organization o that stores data types $D = (d_1, \ldots, d_n)$

about a user U. There is a set of actions $A = (a_1, \ldots, a_m)$ that can be executed on those data, and as a result the data might change, new data might be created, or old data might be deleted.[2] For now we do not assume that data is forwarded (this function will be added in the next subsection).

The access control list $ACL_o \subseteq A \times 2^D$ of an organization o defines the set of data δ items on which each given action a may be performed. We assume that if (a, δ) $\in ACL_o$then a has no undesired impact on any data outside δ. This is a very simplified model for access control, but it is straightforward to add details and constraints if needed. We assume that ACL_o is maintained by organization o. Organization o also maintains a subset $ACL_M \subseteq ACL_o$ which contains all pairs the user cannot block (the actions serving mandatory purposes[3]).

The user specifies a set of blocked pairs $block_U \subseteq A \times 2^D$ and permanently shares this list with o. "Blocking" a pair corresponds to moving the pair into $block_U$ "unblocking" corresponds to removing it from $block_U$. The actual access control list ACL_e that is used for enforcement is derived from the inputs provided, ACL_o, ACL_M, and $block_U$, as follows:

$$ACL_e := ACL_o - (block_U - ACL_M).$$

In this abstract model, the impact of "blocking" a pair (a, δ), i.e., adding it to $block_U$, is the following:

- If (a, δ) $\notin ACL_e$ then there is no immediate impact since the blocked item was either not in ACL_o or else mandatory.
- If (a, δ) $\in ACL_e$ and (a, δ) $\notin ACL_M$ then the pair (a, δ) is removed by updating ACL_e, i.e., o can no longer execute this action on this data.

The impact of "unblocking" a pair (a, δ), i.e., moving it out of $block_U$, is the following:

- If (a, δ) $\notin ACL_o$ then there is no immediate impact. The pair was never in ACL_e or it was in ACL_o but removed by o.
- If (a, δ) $\in ACL_o$ then the pair is re-added to ACL_e.

Note that $block_U$ is not necessarily a subset of ACL_o, i.e., the user might block and unblock a pair (a, δ) before it is actually added to ACL_o. If o adds a pair to ACL_o that is blocked then this pair will not be added to ACL_e. Whenever any of the inputs change, the resulting ACL_e needs to be recomputed. An update is triggered by changes in ACL_o, ACL_M, or $block_U$. If o removes a pair from ACL_M, but not from ACL_o, then this pair can be blocked in the future. If the pair is also in $block_U$ then the update procedure will immediately remove it from ACL_e.

In our abstract model, $block_U$ is an arbitrary subset of $A\times 2^D$. Listing all elements in $block_U$ is the most obvious way to specify $block_U$, but is unlikely to cover many interesting cases. More practically relevant approaches that can be combined are listed below:

[2] Note that purposes can be encoded in actions. If a certain action is a is allowed for one purpose p but not for another purpose p', then action (a, p) and (a, p') would have to be elements of A to enable distinction.

[3] The probably most relevant way to practically define ACL_M is to classify A into mandatory actions A_M and discretionary actions $A_D := A \backslash A_M$, and define $ACL_M := ACL \cap A_M \times 2^D$.

– The all-or-nothing approach can be implemented by setting either $block_U$:= $A\times 2^D$ (everything blocked) or $block_U := \emptyset$ (nothing blocked). Note that this does not block the pairs in ACL_M, which in this case should also include all pairs that are required to perform the unblock operation.
– There might be predefined classes $C_1, \ldots, C_j$ which cover subsets of of $A\times 2^D$, and U can pick any subset of these classes. In this case $block_U$ is the union of all blocked classes. This is similar to the opt-in and opt-out grouping of statements in [12].
– Instead of blocking pairs the user might block data or actions, or classes of data and actions. If the user wants to block all actions in $A*$ and all data in $D*$, then we set $block_U := A^* \times 2^D \cup A \times 2^{D^*}$. The way to deal with classes of actions and data is obvious.

These operations allow users to block portions or all of their data. Note that in a user-centric identity scheme, this blocking/unblocking can be managed by a client-side application. This means that if a user visits a site, the required data is unblocked. Once a user has performed a transaction and logs out, the data is blocked again.

4 Managing Privacy across Multiple Organizations

Organization o will often share data with other organizations. We now discuss the protocols that allow an organization to disclose data to another entity and maintain the privacy of the disclosed data. This includes disclosure and update of data as well as blocking/unblocking.

Traditional privacy policies allow individuals to specify which actions by which organization are allowed on which data elements. They often do not contain explicit disclosure controls, i.e., they do not specify who is allowed to obtain copies of the data [15]. To simplify data handling, we pursue a simpler and more flexible approach. Our concept is split into two parts. *Disclosure control* prevents data from being disclosed to parties that are not trusted by an individual user. *Usage control* then restricts usage and manages block/unblock for trusted organizations that store data of an individual. An organization is allowed to use data if the organization is trusted and if the required permissions have been delegated to this organization.

4.1 Preventing Disclosure to Untrusted Organizations

The best protection of data against an untrusted organization is not to disclose the data to this organization. This means that either an individual trusts an organization sufficiently to act as a guardian of his or her data or else the organization should not obtain the data in the first place. Since this trust depends on the data types, we specify which parties are *in principle* allowed to handle given data items. Those parties are then trusted to enforce the privacy restrictions as specified by an individual.[4] Special care needs to be taken to

[4] This is similar to and can be augmented by the concept described in [10] where parties are only trusted to handle data if they implement a privacy layer that is protected by means of Trusted Computing hardware.

handle the fact that an organization gets the same data via multiple paths. We model this concept as a data-flow matrix DF. Given a set of organizations O and a list of data items D, the data-flow matrix is a subset $DF \subseteq O \times D$ that lists the organizations that are trusted to handle each particular data item.

The corresponding data disclosure can now be specified as follows. Whenever an organization needs to disclose a set D of data to an organization o', it computes a data subset D' such that $d \in D' \subseteq D$ iff (o',d) $\in DF$ and $d \in D$. The recipient should not obtain trust information unless it is trusted to protect the corresponding data. Therefore, the organization computes the subset of the data-flow matrix $DF' \subseteq O \times D'$. The organization o can then disclose D' and *DF'* to its peer o'. If the trust is not sufficient to pass on critical data for a given transaction, the organization can either cancel the transaction or else ask the user to extend the data-flow matrix. The handling of the data-flow matrix will be discussed in more detail in Section 5.2. Note that cryptographic protection can be implemented on top of this scheme, i.e., if a user releases encrypted data (say for o''), the DF specifies who can get hold of and pass on the cipher text. If it includes o'', then o'' can get hold of the data and actually decrypt it. This concept can be used to tunnel critical data (e.g., SSN, credit cards) through multiple semi-trusted parties to the actual intended recipient.

4.2 Managing Data Usage Permissions for Disclosed Data

Permissions are handled along a dynamically generated directed delegation graph that may have cycles. The organizations are the nodes of this delegation graph. Edges correspond to actual disclosures. Each disclosure (edge) delegates a set of permissions that is a subset of the permissions received. An edge is labeled with the data that has been disclosed as well as the corresponding metadata. The metadata consists of the reduced data-flow matrix *DF*, a disclosure history, and the associated permissions. The data-flow matrix defines the maximum set of organizations that can obtain subsequent disclosures; the history defines via which intermediaries the data was received, and the permissions are the associated access control lists as defined in Section 3. Blocking blocks a (subset of) a given edge. An action on data can be performed by a node as long as any unblocked incoming edge still permits this action. In practice this, for example, means that a shipment company can use an address as long as some wholesaler still has an ongoing delivery for this customer. We now formalize this intuition depicted in Fig. 4.

In order to receive disclosures, each organization needs to store such a triple (history, data-flow matrix, permissions) for each received disclosure. Note that an organization can receive the same data via different paths and with different or identical permissions. The organization needs to store and maintain received disclosures separately in order to manage the blocking of a single disclosure.

We now explain the permission handling for disclosures that is summarized in Table 1 in more detail. Permissions of a given organization o are formalized by a triple $(ACL_o, ACL_M, block_U)$ that contains the access control list, the list of mandatory permissions, and the blocked permissions.

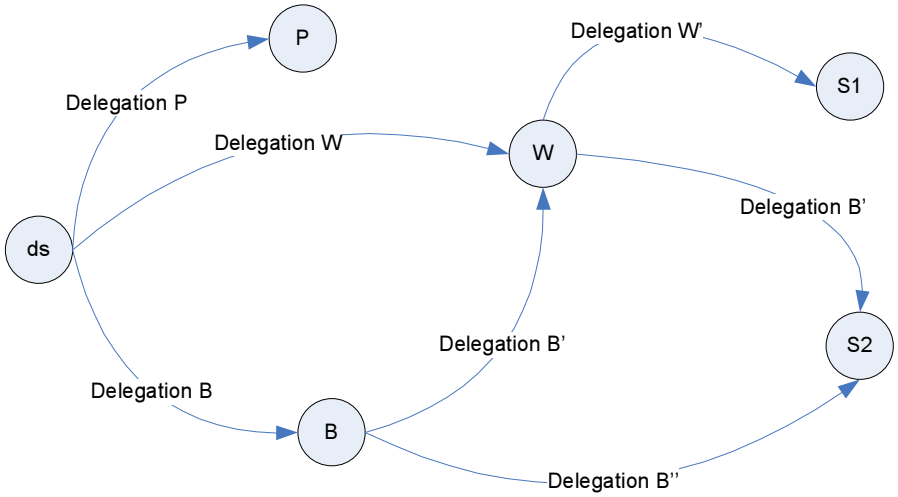

Fig. 4. Dynamic graph of actual delegations for bookstore scenario (S1, S2 are two shipping providers, "ds" is the data subject and edges are labeled with delegated permissions).

Table 1. Computation of metadata for disclosures

Field	**Sender** o	**Recipient** o'	**Constraint**
Data	dat	dat'	$dat' \subseteq dat$ and $(O' \times dat') \subseteq DF$
Data-Flow Matrix	DF	DF'	$DF' \subseteq DF \cap (O \times dat')$
History	$hist$	$hist'$	$hist' := (hist, o)$
Access Control List	ACL_o	ACL'_o	$ACL'_o \subseteq ACL_o$
Mandatory Permissions	ACL_M	ACL'_M	$ACL'_M \subseteq (ACL_M \cap ACL'_o)$
Blocked Permissions	$block_U$	$block'_U$	$block'_U \supseteq block_U$

Disclosure and Delegation: Let us assume that an organization o holds data d with permissions $(ACL_o, ACL_M, block_U)$ received via a single edge, and intends to disclose it to an organization o'. In order to disclose the data, the organization decides on a subset $ACL' \subseteq ACL$ of unblocked permissions to delegate. Both organizations also agree on a set $ACL'_M \subseteq ACL_M$ that defines the mandatory permissions. Once these permissions are defined, the data is disclosed along with the permission vector $(ACL_o, ACL_M, block_U)$, a subset *DF'* of the data-flow matrix that covers the disclosed data, and the history including sender o and recipient o' as its last two elements. The disclosing organization o also stores the history to enable later updates or blocking of the disclosed data. If an organization wants to delegate a data set of data received via multiple edges, it needs to choose one incoming edge for each data item and break the data down into multiple disclosure messages – one for each incoming edge. This is necessary because the incoming data may have different data-flow matrices or different policies associated with the actual data to be disclosed.

Once an organization has received data of a given data subject *ds* via one or more paths, it can compute the actual permissions by combining the permissions along all

edges. The actual ACL is computed as the union of all incoming sets ACL_e that are computed as specified in Section 3. This formalizes the intuition that data can be used as long as at least a single organization has delegated the corresponding usage. The blocking set $block_U$ and the mandatory permissions ACL_M are maintained separately for all in- and outgoing edges. They are considered when updating the individual ACLs and therefore indirectly influence the corresponding ACLs.

Block/Unblock: There are two types of blocking data. A user can either block information via the organization that collected it or else visit the privacy panel of a particular organization directly to block all or portions of the stored data (no matter where the data was collected). The first type is a user blocking data that has been released to an organization o. In this case, the blocking is recursed along the dynamic disclosure path by updating the policies at each edge. If a user has blocked a set $block_U$ at organization o, then o examines along which paths the given data has been disclosed and discloses an updated $block'_U$. The subsequent parties then update ACL_e for the given edge accordingly. Note that this will only block the data if the organization has not received the same data and permissions via another path. The rationale behind this behavior is that in most cases, from a user's perspective different paths are seen as independent. If, for example, a user blocked all data at a given online retailer, the individual would be surprised if the credit card were rendered unusable also for other online retailers. Unblocking again updates the permission sets and propagates the permission update.

In the second type of blocking, where a user visits an organization directly (i.e., clicks on the privacy panel of an inner node and authenticates), the user will review the information stored at this organization and can also see for what the information is used and where it came from. In this case, a user can again block all or portions of the data usages (except mandatory usages). This is then again propagated along the disclosure graph. Note that in contrast to the first approach, this enables the user to block any non-mandatory usage of data at a given organization, regardless of how and where the data was actually collected.

A special case for permissions and block/unblock are circles in the disclosure graph, i.e., that permissions are delegated along loops. We resolve this by keeping the history. This enables the organizations to effectively distinguish original permissions from permissions that are looped-back.

5 Enhancing User Control

The scheme described so far assumes that the parties in the disclosure set *DF* correctly implement the proposed scheme. This includes correct policy enforcement as well as blocking and unblocking. In this section, we investigate how these trust assumptions can be reduced to provide a higher level of verifiability and thus security to the individual end user.

5.1 Increased Transparency

The main benefit of our concept is enhanced transparency. A user can review which data is stored by whom and to which organizations it has been disclosed. By default, an

individual customer will handle data via the party to whom the data has been disclosed initially. This means that the customer can visit one of the previously visited web-sites and review or block/unblock data. This can include browsing along the disclosure graph. In the special case that a customer decides to distrust an organization further along the graph, he or she should directly visit the corresponding privacy panel and then prune the disclosure tree by blocking or deleting a given branch. Any good implementation of our method should provide feedback to the user regarding the precise meaning of a successful or failed block or unblock operation. Since $ACL_o$and ACL_M might change over time, blocking and unblocking can have a delayed impact. The user should be given the option to be informed about such delayed impact before as well as at when the impact actually happens.

5.2 Dynamic Trust Management

An important aspect of our concept for multi-organization privacy protection is the trust management that determines who may obtain which data in the first place. There are multiple options that can be mixed in practice:

- *Enterprise Data-Flow Matrix*: A web-site has a fixed set of subsidiaries that are required for a site to work. E.g., all payments are processed through a given payment processor. In this case, the site proposes a fixed data-flow matrix and the individual customer can accept or decline. This is today's solution.
- *Customer Data-Flow Matrix:* The customer proposes a data flow matrix to the enterprise. The enterprise then dynamically selects providers that are in this matrix. This service-oriented approach requires that the customer has at least one service of each type that is required by a given organization. This needs to be validated by the organization.
- *Dynamic Delegation:* For this approach, there is no complete data-flow matrix initially. The customer (via its federated identity management scheme [14] or individually authorized by a party trusted by the customer) is dynamically asked to delegate certain sub-services to providers that are trusted by the individual. This means that the user points to a trusted payment processing service once needed. Once the organization has obtained permission to disclose certain data, this is dynamically added to the data-flow matrix.
- *Role-based Delegation:* The original matrix only contains roles or groups of organizations, e.g., partners certified by auditor *X*, organizations that satisfy certain privacy requirements, or partners that have signed a privacy protection contract with the user. This concept is similar to the organizations specified by P3P [12]. However, the core difference is that our approach allows later review of the data and the actual organizations that store the data of an individual.

Each of these mechanisms ensures that data disclosure is only permitted to organizations that an individual user trusts. Revoking trust is done by updating the data-flow matrix and the access-control matrix. The consequence is that the corresponding organizations are asked to delete the data they store. If this deletion is not possible legally, one again needs to distinguish between a default and a mandatory data flow matrix *DF*,

where the latter specifies the minimum set of organizations where the data cannot be revoked. Note that the main impact of removing trust is to prevent storage of data corresponding to future transactions. Without additional audits or control, the actual deletion of data cannot be verified in general.

5.3 Verifiable Blocking and Unblocking

The preceding sections assumed that organizations "follow the rules". We now describe a way to increase the auditability of our solution. Our goal is to enable auditors and users to validate that all data that is found at an organization has been obtained legally and has not been blocked. The core ideas are to delete all blocked data, to sign disclosures, to provide receipts for blocking requests, and to require authorization for any data stored at the enterprise.

Our first naïve implementation of unblocking merely modifies the access rights without actually protecting the data. This is a valid implementation approach, but has certain disadvantages:

- Blocking does *not* add a new layer of defense, i.e., this type of blocking does not necessarily reduce the risk of unintentional disclosure.
- Blocked data are of no value to the organization, but the burden of maintaining them is with o. From an economic perspective, a more natural choice would be to put that burden on the individual customer U.

Offloading the burden of maintaining the data to the user can be done as follows: As a result of "block", the organization compiles all data that serves no purpose other than allowing unblock, and send this data token to the user, in some protected fashion (encrypted and authenticated). The user is supposed to store this data, and return it as part of the unblock token. Delayed impact of block might create new tokens, which are all sent to the user. This approach adds a second layer of defense and puts the storage burden on the user. If this storage burden is not considered significant for o, one can use an alternative approach: U can select a cryptographic key and hand the (public) key to o, and o uses that key to protect all this data, i.e., storing the data in its encrypted form without actually knowing the key needed for decryption. Upon unblock, U hands over the (secret) key to o, and o can decrypt all data needed to perform unblock.

In both cases, the user achieves an increased level of control. The most important benefit is that once an honest organization has blocked data, it cannot unilaterally change its mind. This is important in order to prevent that a bankrupt organization sells previously blocked data. Next, we discuss this data token approach as it provides a benefit (and thus an incentive) to the organization that implements this approach. In order to provide accountability for all data that is stored at an enterprise, all disclosures need to be signed (including data, history, and permissions as described above). This enables an enterprise to show that all data has been obtained through a legal disclosure path. When an individual asks an organization to block data, an organization needs to be able to show that it honored the recursive blocking request. This can be done by protecting its own data as described above and requiring blocking proofs from all data recipients. This can be achieved by the protocol depicted in Fig. 5. We assume a confidential channel between the different organizations to prevent data leakage. The customer initiates

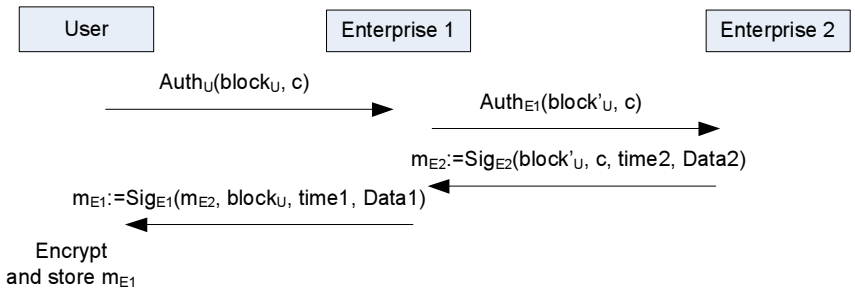

Fig. 5. Example of a blocking protocol

the blocking protocol by sending an authenticated message containing the updated access control list $block_U$ and a challenge c. If the enterprise has not disclosed the data further, it responds by sending a signed response that contains the updated $block_U$' the time from which it promises to no longer use the data, and the only remaining copy of the actual data that has been blocked. If parts of $block_U$ have been delegated to one or more other organizations $E2$, the organization executes this blocking protocol recursively to block and retrieve the corresponding data. Again, the user will obtain a signed blocking receipt from each organization that held data. Note that the blocking message between different organizations is no longer authenticated by the user. Instead, our concept enables all organizations to block delegations they have previously performed. In the protocol this implies that subsequent blocking messages are authenticated by the sending organization. Furthermore it implies that these blocks can contain a subset of $block_U$ if only a subset has been disclosed. Both enterprises then send signed messages with the blocking time to the user. The user encrypts and stores the data.

To unblock the data, the user returns the blocking token m_{E1} (and m_{E2}) and an authenticated message that allows the organization to unblock the data. The organization then recursively recovers the data from this backup. Note that the authorization to unblock is needed to enable the enterprise to prove that the unblock operation was legitimate, i.e., that it did not cheat the user by sending a blocking token while continuing the usage of the data.

Usually the data used to authenticate U is part of the data protected by our concept (i.e., part of D), and authenticating U is implemented through one or more of the actions in A. To facilitate unblocking it is appropriate to add all pairs (a, δ) representing authentication to the set ACL_M, i.e., to prevent U from accidentally blocking authentication as long as other data can still be blocked. Another approach is to allow blocking of the identity and authentication information. In this case, the user would be required to unblock its identity to perform additional block operations.

Note that without additional assumptions it is impossible to achieve complete verifiability against an organization that is completely untrusted. Known assumptions that enable verifiable deletion are a limited amount of memory [13], or the assumption that data is stored only in a well-defined trusted infrastructure [3], or an infrastructure that supports secret sharing with at least one honest storage server [16].

5.4 Protection of Mandatory Data

In general, data that is mandatory cannot be deleted or blocked. This is either caused by pending transactions that need the data or else by regulatory requirements. In the first case, blocking would terminate the transaction and usually is not desirable. In the second case, usage can be limited to certain well-defined operations such as an annual audit, or following a court order.

We now sketch additional means to offer similar protection for mandatory permissions. The core idea is that the limited amount of mandatory data can be protected by mutually trusted entities. From the enterprise's perspective, this trustee guarantees the legally required availability of the data. From an end-user perspective, the entity ensures that the usage is limited to the pre-defined purposes. Potential options for implementing these trusted third parties are secure hardware (smartcards, secure coprocessors), additional trustees (either for secret decryption keys or the data itself), or special storage devices such as a secondary write-only data store that only reveals data under the conditions specified. One example are log-files that are kept for auditing purposes. In this case, the data can be public-key encrypted and the decryption key is to be escrowed between the user, the organization, and a third party. If the legal pre-conditions are met, the user pre-authorizes this third party to release its key share to the organization in order to meet the regulatory requirements.

6 Conclusion

We have presented a simple concept for increasing transparency and control over the use of data for individual end-users. The core idea is a standardized privacy panel that allows users to review the data that has been stored about them. In contrast to earlier approaches, we describe how to enable temporary suspension of personal information. This blocking renders data unusable while a certain service is not in use. This protects the individual from accidental misuse. Furthermore, we separate data-flow restrictions from data usage restrictions. The rationale is that data should only be disclosed if the recipient is trusted to enforce the associated policies.

Acknowledgments

We thank the anonymous reviewers for valuable comments that enabled us to substantially improve the final version of this paper. We are grateful to Charlotte Bolliger for improving the readability and presentation of this paper. Finally, we would like to thank our colleagues Mike Nelson and Jane Johnson for helpful discussions on privacy in service compositions. This work was partially supported by the OpenTC project www.opentc.net that is funded by the 6th framework programme of the European Commission.

References

1. Lorrie Cranor, Marc Langheinrich, Massimo Marchiori: A P3P Preference Exchange Language 1.0 (APPEL1.0); W3C Working Draft, 15 April 2002, http://www.w3.org/TR/P3P-preferences/.
2. Michael Backes, Walid Bagga, Günter Karjoth, Matthias Schunter: Efficient Comparison of Enterprise Privacy Policies; 19th Annual ACM Symposium on Applied Computing, Nicosia, Cyprus, March 14-17, 2004, 375-382.
3. G. Badishi, G. Caronni, I. Keidar, R. Rom, G. Scott: Deleting Files in the Celeste Peer-to-Peer Storage System; 25th IEEE Symposium on Reliable Distributed Systems (SRDS '06), IEEE Press, October 2006, 29-38.
4. Adam Barth, John C. Mitchell: Enterprise Privacy Promises and Enforcement; Proceedings of the 2005 Workshop on Issues in the Theory of Security, Long Beach, California, ACM Press, 2005, 58-66.
5. Michael Backes, Birgit Pfitzmann, Matthias Schunter: A Toolkit for Managing Enterprise Privacy Policies; 8th European Symposium on Research in Computer Security (ESORICS 2003), LNCS 2808, Springer-Verlag, Berlin 2003, 162-180.
6. David Chaum: The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability; Journal of Cryptology 1/1 (1988) 65-75.
7. Roger Dingledine, Nick Mathewson, Paul Syverson: Tor: The Second-Generation Onion Router; Proceedings of the 13th USENIX Security Symposium, August, 2004.
8. Günter Karjoth, Matthias Schunter, Els Van Herreweghen: Enterprise Privacy Practices vs. Privacy Promises - How to Promise What You Can Keep; 4th IEEE International Workshop on Policies for Distributed Systems and Networks (Policy '03), Lake Como, Italy, June 4-6, 2003, 135-146.
9. Günter Karjoth, Matthias Schunter: A Privacy Policy Model for Enterprises; 15th IEEE Computer Security Foundations Workshop (CSFW), IEEE Computer Society Press, Washington 2002, 271-281.
10. Michael Kinateder, Siani Pearson: A Privacy-Enhanced Peer-to-Peer Reputation System; 4th International Conference on Electronic Commerce and Web Technologies (EC-Web 2003), LNCS 2738, Prague, September 2003, 206-215.
11. Stephen E. Levy, Carl Gutwin: Improving Understanding of Website Privacy Policies with Fine-Grained Policy Anchors; WWW 2005, May 10-14, Chiba, Japan, ACM Press, 2005.
12. Lorrie Cranor, Brooks Dobbs, Serge Egelman, Giles Hogben, Jack Humphrey, Marc Langheinrich, Massimo Marchiori, Martin Presler-Marshall, Joseph Reagle, Matthias Schunter, David A. Stampley, Rigo Wenning: The Platform for Privacy Preferences 1.1 (P3P1.1) Specification; W3C Working Group Note 13 November 2006, http://www.w3.org/TR/2006/NOTE-P3P11-20061113/.
13. Andreas Pfitzmann, Birgit Pfitzmann, Matthias Schunter, Michael Waidner: Trusting Mobile User Devices and Security Modules; Computer 30/2 (1997) 61-68.
14. Birgit Pfitzmann, Michael Waidner: Federated Identity-Management Protocols; 11th International Workshop on Security Protocols (2003), LNCS 3364, Springer-Verlag, Berlin 2005, 153-174.
15. William Stufflebeam, Annie I. Antón, Qingfeng He, Neha Jain: Specifying Privacy Policies with P3P and EPAL: Lessons Learned; Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society (Washington DC, USA, October 28 - 28, 2004). WPES '04. ACM Press, New York, NY, 35-35.
16. Adi Shamir: How to Share a Secret; Communications of the ACM 22/11 (1979) 612-613.