Privacy Policies Compliance Across Digital Identity Management Systems

Anna C. Squicciarini
College of Information Sciences and Technology
The Pennsylvania State University
University Park, PA, USA
acs20@psu.edu

Alexei Czeskis
Department of Computer Science
University of Washington
Seattle, WA
aczeskis@washington.edu

Abhilasha Bhargav-Spantzel
Intel Corporation
Santa Clara, CA
abhilasha.bhargav-spantzel@intel.com

ABSTRACT

An emerging approach for protecting identities of individuals, while at the same time enhancing user convenience, is represented by *federated identity management* systems. In this paper we develop an approach to support privacy controlled sharing of identity attributes in federated environments. We present a wide range of strategies that enable users to trace their personal information across the federation and verify whether it has been managed according to their privacy preferences. Users can employ one or more of these strategies according to their goals and priorities. Additionally, we analyze the challenging issue of data, privacy policy, and preference updates in a federated system. Our algorithm allows users and federated service providers to control whether new versions of data and users' privacy references have been used, and detect possible inconsistencies.

Categories and Subject Descriptors

K.6.5 [**Computing Milieux**]: Management of Computing Information Systems

General Terms

Security

Keywords

Privacy, Digital Identity,Tracing

1. INTRODUCTION

As activities such as shopping, discussion, and transactions are more and more often conducted in the cyber world, digital identity management has become an integral part of today's organizations issues. An emerging approach for organizations to manage individuals' identity attributes at an inter-organizational level is based on the no of Federated Identity Management. Federated identity managements systems (FIM, for short) enable organizations to provide services to qualified individuals; and empower them with control over the usage and sharing of their identity attributes within the federation[11, 10, 18]. A FIM system consists of software components and protocols that handle the PII of individuals throughout their identity life cycle. It involves three main types of entities, namely the user, identity provider (IdP) and service provider. The IdPs manage and provide user identities and potentially issue user credentials, and the service providers (also known as relying parties) are entities that provide services to users based on their identity (represented by PII). As a matter of fact, the default approach in FIM systems is for federated service providers (FSP) to share PII upon request, without involving the IdPs.Users are typically affiliated with one specific FSP to whom they submit PII at the time of registration. They also disclose other PII to gain access to specific services or data made available by the other FSPs.

When collecting PII data, users often express privacy preferences to FSPs regulating the distribution and use of the PII. However, simply because FSPs advertise their privacy promises, and collect users' preferences, there is no guarantee that they are providing good privacy protection with respect to the collected PII. In fact, FSPs are not required to have the privacy technology to enforce their privacy policies.

The voluminous exchange of personally identifying information (PII) has resulted in privacy violations being a common occurrence today, even from well established companies [12]. One approach that is emerging to overcome such privacy issue is that of user-centrity[4], where users' involvement is required each time any user's data disclosure occurs. This approach is not convenient for end users, who may not always be interested in controlling every action involving their data. A more suitable solution would be for users to be able to trace their data and check FSPs privacy practices as needed.

Developing a light-weight privacy preserving methodology, by which users and FSPs can track PII and related privacy practices in a FIM is a not trivial task. Conventional monitoring approaches are not suitable; traditional enforcement mechanisms would in fact imply persistently monitoring users' transactions and collecting their pri-

SPRINGL 2008, November 4th, 2008, Irvine, CA, USA.

vate information, which goes against the philosophy of a privacy preserving solution. Mechanisms of this type would result in a system where stored context data could grow exponentially as the rate of interactions or number of users in a FIM increase. Additionally, securing communication channels and encrypting messages may help preserve the privacy of relevant information only up to some extent. Such solutions are usually unpractical, as they require users and FSPs to manage cryptographic keys, and do not provide users with the capability of locating their data.

Complexity is added by the fact that users' personal data, as well as their preferences, can (and often do) change over time. These changes are often prompted by FIM entities' decisions (such as newly adopted practices) or because of external events, such as expiration of certificates (a gift certificate, a phone card), revoking of data (e.g. a credit card number), or change of content (e.g. a user change of work address). The flexibility provided by versioning of data and privacy preferences can introduce inconsistencies into the FIM system. Inconsistencies may have a serious impact, with consequences ranging from unintended data disclosure to lack of compliance with users' privacy preferences and errors in the users' PII usage. We believe inconsistencies in FIM are important yet underestimated problems in the area of privacy management, which can lead into serious privacy breaches.

In this paper we propose a methodology to address the above issues. Specifically, we propose tracing mechanisms that make it possible for users to trace their PII across a federated system and verify whether their data has been managed according to their privacy preferences. Since users may be interested in monitoring their PII for different reasons, in the face of temporal and computational constraints, one single approach to perform tracing may not be adequate. As such, we present a number of different strategies that users can select according to their goals and priorities. Our approach entails a form of accountability since an entity non-compliant with the users' original privacy preferences can be identified. We also discuss how these techniques can be applied to specific domains characterized by a broad disclosure of sensitive information across federated domains. Second, we show how our techniques take into account the challenging issue of PII and user's data and privacy preferences updates by developing a notion of consistency specific for FIM, and a related consistency checking algorithm. This algorithm allows users and FSPs to control whether new versions of users' data and privacy policies have been used and detect possible inconsistency. To the best of our knowledge, no previous approaches exist that provide several strategies for privacy policy tracing and deal privacy preferences and data versions in FIM systems.

The remainder of the paper is organized as follows. We first present a summary of the main concepts related to privacy policies and policy subsumption. Next, we present the tracing algorithm and the main strategies supported by our solution. In Section 5 we present the main usage scenarios for our solution. In Section **??** we analyze consistency and propose a solution for detecting inconsistency. We discuss related work in Section 6; and conclude the paper in Section 7.

2. PRELIMINARY NOTIONS

Our approach relies on two important notions of privacy policy matching and privacy policy subsumption. In what follows we provide background information about these notions that is relevant for the subsequent discussion in the paper.

2.1 Notations and main assumptions.

FSPs are collaborative entities, that are willing to share users' received data according to the contractual agreements amongst the FIM members. Each of them publish privacy policies that are made available to users and other FSPs in the federation. There are however no automated methods to monitor the actual FSPs privacy practices. However, we assume that each FSP must maintain a tamperproof accounting/monitoring system which keeps track of its privacy related information in terms of policies published and subsequent updates[1]. Specifically, we rely on the following assumptions: (1) the log relation resulting from the monitoring mechanism is horizontally partitioned among the FSPs involved in the recorded events, (2) the logging is done synchronously with the occurrence of the FSP entities interactions, and (3) FSPs and users have access to such information, which is either locally stored at the federated members or maintained at their own independent systems; depending on the specific settings and trust existing among the federated members.

We assume that a domain independent ontology exists that includes concepts for data, actions, privacy terminology, and so forth. In order to reason about privacy policies (policies, for short) - without having to deal with various speci■c syntaxes, we rely on a compact notation of privacy policies and preferences which is abstract and independent from any existing lower level language. The main symbols are reported in Table 1.

2.2 Privacy Policy Subsumption

In order to streamline daily tasks, FSPs within a FIM can request users' data from other FSPs with which the users have previously interacted. For example, in a medical environment, authorized personnel of a clinic can share medical records with colleagues of federated clinics to define a patient's eligibility to a given treatment or to evaluate his health condition.

To enable secure information sharing across FSPs, the privacy policies of every FSP that receives information pertaining to a given individual should comply with the individual's original preferences. Compliance can be verified either by directly checking users' privacy preferences with the FSP's policies or by performing policy subsumption [20] checks. Policy subsumption allows FSPs to automate the process of checking for policy compliance when transferring data from one FSP to another. Subsumption reasoning is used for policies defined over equal or similar classes of data in order to determine if they conflict. Specifically, the process for determining the subsumption of two policies is executed between two FSPs when one FSP (referred to as FSP1) requests one or more

[1]Notice that this assumption is realistic in that service providers are required to maintain logging information about customer's PII and make it available for auditing[23]

Notion	Symbol	Variables	Encoding
Privacy Policy	$PPol^d_{FSP:t}$	d: data the policy refers to, t: time of publishing and FSP the policy publisher	declarative format, using languages such as P3P [6];
Privacy Preferences	$PPref^d$	d data the preference refers to	XML based languages such as [14] or [1].
Matching	$PPol^d_{FSP:t}$ $\mathcal{M}$ $PPref^d$	d data for which matching is executed	Privacy compliance tools[**?**, **?**]
Subsumption	$PPol^d_{FSP1:t} \prec PPol^d_{FSP2:t}$	$PPol^d_{FSP1:t}$: Subsumed policy $PPol^d_{FSP2:t}$: Subsuming policy	Policy subsumption for data d [19]

Table 1: Notation symbols

user attribute from another FSP (referred to as FSP2) and is denoted as $PPol^d_{FSP1:t} \prec PPol^d_{FSP2:t}$. The process varies depending on whether FSPs rely on standardized *policy templates* or on *customized policies* to specify privacy practices.

- A number of standardized privacy policies to which all federated entities can refer to is available, as suggested by the Liberty Alliance framework [11]. Each FSP can choose a policy template T_i from the set $\{T_1; .....; T_n\}$ of available templates. The templates in this set are totally ordered based on the strictness approach that will be followed for data disclosure. In order for information to be released between two FSPs, the associated policies must be compatible, that is, if data is being released from FSP2 to FSP1, then policy enforced by FSP1's policy should be equal or *stricter* than the policy applied by FSP2.
- Customized policies allow FSPs to autonomously specify their policies and match them for subsumption upon data sharing among FSPs. In this case, the matching process is more articulated. Specifically, to evaluate the relationship between two given policies, *PPol2* and *PPol1*, associated respectively with the requester and the data holder, all the values assigned to policy components must be analyzed [19]. Typically, according to the P3P[6] specification, these correspond to elements such as *purposes, recipients,* and *retention* for all data being requested from *PPol1*. Therefore, every data element that is being requested by holder of *PPol2* is evaluated to determine whether the requester's intended use of the data element is subsumed by those in *PPol1*.

3. STRATEGY BASED RESOLUTION PROTOCOLS

Automated transferring of data among FSPs, without direct user involvement, may result in wide distribution of users' data across the federation. In order to verify that the data has been distributed in compliance with the original intention of the user, we propose a *tracing* protocol and several alternative *resolution* algorithms for performing the tracing.

3.1 The Tracing Algorithm

Our solution for tracing is accomplished by an extended version of the basic *Policy Tracing* algorithm [20], illustrated in the algorithm reported in Appendix A. Essentially, the tracing algorithm works by checking subsumption of privacy policies iteratively based on our assumption that the transcript of the matching is stored in a trustworthy manner on a tamper-proof device at each FSP.

Specifically, the tracing is accomplished in the following manner:

– Every unused FSP, FSP_u determines (based on local audit logs) the set of all *child* FSPs $= \{FSP_1, \ldots, FSP_i\}$ to which FSP_u passed the user's data

- For every child $FSP_c \in \{FSP_1, \ldots, FSP_i\}$, use their local audit logs to discover all *grandchild* FSPs $= \{FSP_j, \ldots, FSP_k\}$ to which FSP_c passed the user's data.
  - For every grandchild $FSP_g \in \{FSP_j, \ldots, FSP_k\}$, select the *most recent* policy of FSP_c, for the given data and check if $PPol^d_{FSP_c:cT} \prec PPol^d_{FSP_g:cT}$ hold true, with FSP_g's policy valid at current time cT. If the policies do not subsume (computed as described in Section 2.2), then the policy versions used during the actual data sharing at a given time T (as opposed to the *current* policies) are used – the current version of policy may differ from the one used at time T. If $PPol^d_{FSP_c:T} \prec PPol^d_{FSP_g:T}$ does not hold true as well, then the algorithm determines this as a point of mismatch. Otherwise, we have verified that FSP_c has adhered to the user's preferences, and we add FSP_c, as an unused FSP, to the trusted set.

– If there are no unused FSPs in the trusted set to explore, then the trace is verified with correct policy harmonization in the whole trace. Otherwise, the tracing is continued on one of the unused FSPs in the trusted set. Note that to bootstrap this process, the initial trusted set must consist of the user. The user must complete the tracing steps to verify FSPs to which he directly passed his data and then add them to the trusted set.

This approach requires cooperation from all FSPs, and is taken as one single FSP cannot be aware of the whole path the user's data has taken throughout the federation. Each FSP only knows the FSP(s) that directly shared the data to, and can retrieve the information about the FSPs to which the latter passed the data on to. This method is resistant against single malicious FSPs and can be re-

verted so that the trace be carried out by one single FSP if colluding FSPs are suspected, as discussed in Section 4.

3.2 Resolution Strategies

The key aspect of the tracing algorithm of above is deciding which *path* to take in exploring the network of FSPs in the federation. The path denotes the sequence of FSPs to explore; and in case of multiple alternatives, it may influence the overall tracing outcome. For example, in Figure 1, from FSP2 there are multiple FSPs reachable. Different tracing strategies can be devised, to help optimize the search according to the trace's main goals and existing constraints. Specifically, by tracing strategy, we mean the approach a FSP adopts (on behalf of the user) in selecting the path to follow, exploiting user's and FSPs knowledge about the FSPs in the federation and their respective trustworthiness.

Our framework supports three basic general-purpose strategies that reflect different approaches to the tracing. In addition, the system provides a mixed strategy which results from the combination of the basic ones. We discuss the strategies in the following, and provide a summary of their main features in Table 2.

Priority Based Resolution Strategy. This strategy's main goal is to carry on an optimized trace, where the order of the FSPs to explore is prioritized. The strategy, reported in Appendix A, selects the FSP with the highest priority by invoking function $HighestPriority(FSP_{dests}, UserIn)$. The function selects FSP's based on the prioritization of different criteria, related to trustworthiness of the FSP and of the data to follow. The resolution priority is inversely proportional to the trust level. Trust-based criteria are defined by means of FSP's possessed credentials and/or reputation values. Additionally, the selection criteria may include the specific user's priorities, and be transmitted throughout the trace by means of the $UserIn$ parameter, and incorporated in the priority decision as meta data.

Such criteria can be used exclusively or in conjunction. In case both the user's priority preference and the FSP's priority order are considered, then a methodology inspired by the prioritized Reiter defaults [3] method is suggested. The Reiter default is of the form: $\frac{\alpha:\beta}{\gamma}$ where α, β and γ are formulae of first-order logic. The formula α is the prerequisite, β the justification and γ the consequent. A default rule can be read intuitively as: *if I can prove the prerequisite from what I believe, and the justification is consistent with what I believe, then add the consequent to my set of beliefs.* In our approach trust can be combined with the priority in such a way that if the FSPs can establish a certain level of trust (β) with specific credentials (α) then the priority (γ) can be deduced.

EXAMPLE 1. *Consider two FSPs* E-Book *and E-Pay. E-Book shares user data with E-Pay and wants to compute the resolution priority of E-Pay. E-Pay has two credentials namely; 1) it is a part of Better Business Bureau and 2) has a partnership contract with E-Book. E-book's policy assigns these two credentials (α) of E-Pay to trust level 4, given that the maximum trust level is 6 (β). This results in a resolution priority of E-Pay equal to 30%.*

After each successful subsumption check, the FSP is flagged as passed (using function $flagFSP\text{-}passed$ in step 20 of

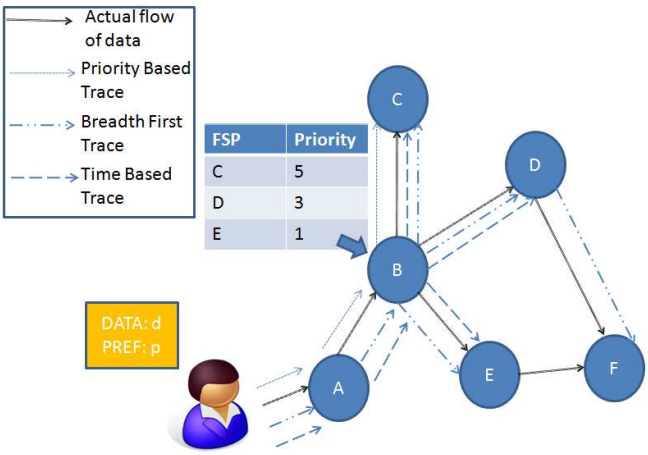

Figure 1: Example scenario of FSP trace

Algorithm **??**). A *flag* consists of the meta data associated with a given trace. Such data includes the data version and the policy version related to that trace. If the trace reaches a flagged FSP then the trace is terminated.

EXAMPLE 2. *In Figure 1, the user has provided FSP A with his data d. A provides it to FSP B, which, in turn, provides it to FSPs C, D and E. At possibly different times, both D and E provide FSP F with the same user data. When A launches the priority based resolution, it first checks B to verify the subsumption of the policies concerning the data d. Then FSP B uses its priority table to determine the direction of the trace. C has the highest priority, therefore the final resultant trace is A-B-C.*

The termination of the trace is based on three conditions. The first two conditions are common to each strategy, and indicate the end upon a mismatch or the lack of FSPs to explore. The third one is strategy specific, and implies that the trace has reached a FSP that has no further destination FSPs with high priority.

Breadth first strategy. The breadth first strategy consists of spanning the trace to all possible FSPs. In this case, for every source FSP, all the possible destination FSPs are checked in a breadth first manner for verification. Verification are first performed in a particular level of the breadth-first tree, before moving on to the expansion of FSP possibilities at the following level. Specifically, the breadth first strategy main steps are summarized as follows.

- The user is the root node of the breadth first trace tree.
- Each FSP_i that the user has provided with data is selected. Given a FSP_i, which would be the source FSP, the destination FSP, FSP_{i+1}, that it has shared the user's data with is determined. If more than one FSP destination is found, all the destinations are considered.
- The subsumption algorithm is performed with all the destination FSPs. If the policy subsumption is successful then the FSP is flagged as *passed*.[2]

[2]Similar to the previous strategy, if a flagged FSP is

Strategy	Selection Criteria	Termination	Time Complexity
Priority Based	1. Trust/Reputation of peers; 2. Users' guidelines; 3. Reiter Defaults	(1) No further FSPs (2) The trace loops back (3) No high priority FSP.	n*c
Breadth first	From FSP_{rec} to each FSP_i that the user has provided with data	(1)	n*c (worst case); c (best case)
Time based	Temporal Branch-on-Need Tree	(1) (2) and (4) Time out	
Hybrid	Combination of above	(1) (2) (3) (4)	n*c or Time out

Table 2: Strategies comparison. n denotes the number of nodes in the FIM, while c is the time to compute subsumption.

- All the destination FSPs are iteratively found until the trace is terminated. Termination occurs as per the general tracing algorithm (see Table 1). If the strategy ends successfully, any two FSPs in the federation system, FSP_i, FSP_k have matching policies.

This strategy is complete, in that all possible FSPs are eventually examined, although it requires the complete trace tree to be constructed during the search and saved for the algorithm to run. In terms of space complexity, the strategy may be computationally expensive if the federation system is composed of several FSP nodes and the traces expand through numerous branches within a single trace. In terms of time complexity, because of the parallelism of the verification, the strategy may be very efficient, if the number of parallel nodes to explore is high. The fastest execution time is c (c being c the time for subsumption) if all the FSPs can be analyzed simultaneously.

EXAMPLE 3. *Consider the FSPs A - F in Figure 1. The breadth first trace starts with the user and the FSP A. User first verifies that A is compliant with the users privacy preferences. As a next step A checks for subsumption with B. In the case of FSP B, the subsumption with C, D and E has to be performed before the depth of the trace is increased. Considering that the subsumption are successful, FSP D checks for subsumption with F before FSP E. In this case F is not checked more than once. When the D–F subsumption occurs, the FSP F is flagged. This flag is verified at the next step of the trace and E does not have to perform any further checks.*

Timing based strategy. The main goal of this strategy is to traverse the best possible trace in a fixed amount of time. To execute the timing based strategy, we use an approach inspired by the Temporal Branch-on-Need Tree (T-BON) [22]. As a first step, the source FSP determines the possible destination FSPs. Then for each destination FSP, a policy match query is constructed. Queries are of the form *(t; sourcePPol; sourcePolicyVersion)*, where t is the time allotted for this trace and *sourcePPol* the source privacy policy to verify. If the match is performed successfully in time t', the destination FSP subsequently becomes the source FSP and sends a query to its destination nodes with time allotted as $t - t'$. This process is repeated recursively until the final FSP is reached or a timeout, that is $t - t' < 0$ condition is reached. In any execution of the timing based strategy the amount of time is evenly distributed amongst the possible destination FSPs.

EXAMPLE 4. *In Figure 1, the user allots FSP A 10 seconds to complete the trace. A and B take 1 second to check subsumption. Then FSP B which has total of 9 seconds, assigns 3 seconds each for each possible sub trace. The B–C, B–D and B–E subsumption checks occur in 1 second each. Say F is not an efficient FSP and it takes a minimum of 3 seconds to complete the subsumption check with it. Therefore as D–F or E–F requires at least 3 seconds to carry out the subsumption check the trace, the trace is not able to check subsumption with F.*

Hybrid strategy. A hybrid strategy is employed using the combination of two or more resolution strategies. The combination of different resolution approaches can relax the strict rules followed by the underlying strategies. The following is an example of hybrid strategy combining the priority based resolution and the timing based resolution.

EXAMPLE 5. *In Figure 1, the user allots FSP A 10 seconds to complete the trace. 'A' and 'B' take 1 second to check subsumption, after which FSP 'B' uses its priority table to assign the amount of time for each sub trace. It allots C and D 4 seconds each and E 1 second. The B–C, B–D and B–E subsumption checks occur in 1 second each. D–F requires 3 seconds. In this case since D has 3 seconds to complete this the A-B-D-F trace can be completed in time.*

The hybrid strategy's main steps are presented in Appendix, Algorithm 2. It is assumed that the priorities at all FSPs and the time allocated for the trace are specified at the beginning of the trace. The key difference, as compared to the priority based trace strategy, is the recursive function HYBRID TRACE. At each point in the trace the total remaining time to conduct the trace is recursively distributed amongst all possible FSPs based on the priority assigned to the destination FSPs. Similar to Algorithm 3, at each hop, the subsumption of the policies of the receiver and destination FSP is carried out. In this trace, the higher priority FSPs can be reached more likely than the rest in the allocated time ($TimeAlloted$).

Strategies Properties.

reached again during the breadth first trace, then the policy match is not performed again.

The proposed tracing strategies ensure *correctness* and *termination* properties. *Correctness* in our context refers to the fact that, if a mismatch exists, the tracing will find it. This property is guaranteed for the visited nodes. It results as a natural consequence of the subsumption verification. *Termination* is guaranteed by each strategy, as per the termination conditions described for each. In the case of priority based search, the termination is guaranteed by the end of the complete trace or detection of loops. For the breadth first and timing based strategy, the depth of the search and the time threshold ensure the end of trace respectively.

4. SECURITY ANALYSIS

We analyze security of our protocols, by focusing on the policy tracing algorithm. The security considerations apply to the resolution strategies as well, since the basic tracing mechanism is maintained by each strategy.

The policy tracing algorithm is resistant to several types of attacks. Specifically, we address the cases of semi-honest, single malicious, and colluding malicious parties. By semi-honest, we mean parties that will follow the tracing protocol, but try to learn as much information as they can during the interaction. Malicious parties, in our case, correspond to parties that have not performed the local subsumption matching correctly and have released user data to parties whose rules for use of data violate the data owner's privacy requirements. Malicious parties will attempt to circumvent being caught. Colluding malicious parties are multiple parties who can freely exchange user's data, possibly violating the data owner's privacy requirements. We assume that one colluding party will never reveal information indicating policy violation by any other colluding party. In case of the strategies, the possibility of detecting the malicious party is constrained by the nodes explored, if the strategy is not complete and the visited nodes bounded by some temporal or prioritization criteria.

The following cases highlight the security features of the tracing algorithm:

All semi-honest parties. If all the parties are semi-honest, matching of FSP's policies in the tracing will be accomplished successfully. This is because of the subsumption verification performed at each step when the data is released between any FSP's.

A single malicious party. A single malicious party in a trace is identified efficiently by a single execution of the tracing algorithm. The tracing begins at the source FSP – the one to which user originally released his/her data. The user bootstraps the tracing by verifying compliance of his preferences, that is if $PPol^{d}_{FSP:t} \ \mathcal{M} \ PPref^{d}$ holds true. The source FSP will continue to execute the tracing algorithm until the malicious party is found. More formally, let us assume FSP_{k-j}, $j < k$, be the malicious party and FSP_k be the source FSP. As the trace continues a recursive procedure is called such that FSP_k checks $PPol^{d}_{FSP_{k-1}:cT} \prec PPol^{d}_{FSP_{k-2}:cT}$, FSP_{k-1} checks $PPol^{d}_{FSP_{k-2}:cT} \prec PPol^{d}_{FSP_{k-3}:cT}$, and so on. This procedure continues until $\text{FSP}_{k-(j-1)}$ is reached. Note that till this point, each party's transcript is checked by an honest party before it is delegated the task of checking its parent. Then as $\text{FSP}_{k-(j-1)}$ checks FSP_{k-j}'s transcript and finds an error, it will send a GM_TRACEFAIL message towards the sink such that the resulting error is notified at the source.

Non-consecutive malicious parties By non-consecutive malicious parties, we mean two or more malicious FSPs that are not consecutive; they did not pass data directly to one another. Specifically, FSP_i be the first malicious party encountered during the trace and FSP_j be the second malicious one, with $i < j$. If FSP_i, FSP_k are non-consecutive, there is at least one non-malicious party FSP_k such that $i < k < j$ and the data flows from $FSP_i \rightarrow FSP_k \rightarrow FSP_j$. Finding that FSP_i is malicious reduces to the *single malicious party* case, hence it will always be found. Once $PPol^{d}_{FSP_j:cT} \prec PPol^{d}_{FSP_{j+1}:cT}$ is found not to be true, the tracing algorithm is restarted at FSP_{j+1}. Note that the tracing must be bootstrapped by the user, who verifies $PPol^{d}_{FSP_{j+1}:t} \ \mathcal{M} \ PPref^{d}$.The case when FSP_{j+1} is malicious falls into the consecutive or colluding malicious parties case. Otherwise, $FSP_{j+1} = FSP_k$ and and FSP_{j+1} is trusted to resume the trace in search of the next malicious party.

Consecutive or colluding malicious parties The presence of colluding parties can be detected if at the end of the trace, the last FSP's policy is not subsumed by the user's policy, and after the trace no GM_TRACEFAIL message is propagated to the user. In this case we require the transcript verification run at each node by a trusted third party (TTP) in a brute force manner. This TTP can also be the user himself. TTP follows the same trace protocol with the only difference that it does the verifications. Once an error is found, the GM_TRACEFAIL message is propagated exactly the same as the original protocol.

5. APPLICATIONS OF THE TRACING STRATEGIES

The availability of a number of tracing strategies empowers users to determine how to trace the data distributed across the FSP. As illustrated in the previous section, the proposed strategies differ in the type of messages to be exchanged and achieve different levels of assurance. To show how they can be usefully adopted in the different scenarios that may occur, we highlight some interesting application cases.

General applications cases.

– Determining which FSPs are currently in possession of the user's data. Since the tracing algorithm can traverse the whole graph which represents the releases of the user's data, the user is able to identify all FSPs that have possessed his data.

EXAMPLE 6. *Suppose Bob wants to know if his data has been released to any FSPs not within his native country. Here U-State would use the priority based strategy and specify that* FSP.location != USA[3]*. The breadth first strategy may be employed if the user wants to have the most possible complete understanding of what type of FSPs*

[3]We assume the information on geographic location to be available for query.

currently have his data or if he wants to discover where his data was released by a particular set of FSPs. Furthermore, Bob can utilize the hybrid strategy to generate meaningful queries concerning the whereabouts of his data.

– Establishing whether user's data has been spread throughout the federation in accordance with his privacy preferences, and (if it has not) identify the guilty party.

EXAMPLE 7. *Alice finds that her data is known at FSP3. As an initial step, Alice checks if her privacy preferences matched the policies of FSP3. The policies do not adhere to her preferences. Thus, some FSP in the federation leaked her data violating her privacy preferences. In order to determine to whom Alice's data was incorrectly leaked, the tracing algorithm is started by FSP3. First, Alice uses the priority strategy based on the least trusted FSPs. Since the guilty party, if it exists, is more likely to be one of the less trusted FSPs, it will be found faster by using this approach. As she is not satisfied with the outcome or she wants to further investigate, she uses the timing strategy. The timing strategy can help to restrict her search to a reasonable amount of time that she is willing to wait for her inquiry to complete. Finally, she uses the breadth-first strategy to discover all of the FSPs to which her data was leaked.*

– Predicting possible release vectors, that is - how the user's data might spread through the federation in the future. The list of all possible release vectors consists of the set of FSP that could possess the user's data such that the user's initial privacy preferences are preserved. This is accomplished by taking the list of FSPs that have already received the user's data and determining to which other FSPs they can release data based on a possible subsumption match. Statistical methods can be employed to determine which data releases have the highest probability. This can be accomplished by analyzing prior interactions between various service providers. These lists of possible release vectors may be employed by the user as a diagnostic to determine if his privacy preferences are too strict or too lax.

Inconsistency detection.

Our strategies can be successfully used to detect privacy and data inconsistency, as well as preventing them by selectively distributing data and privacy preferences updates to the interested FSPs.

In our context, consistency means that privacy policies should be compliant with users' preferences, despite potential changes in users' data, preferences and/or policies, and the proper policy and data versions used at any time. Consistency can be either evaluated locally (at a single FSP) or globally in the whole FIM. Global consistency requires having the same version of the same data for all users within the FIM. It also implies that all federated entities be aware of the most current data and privacy preferences, if multiple versions exist. To achieve global consistency, the conditions for a notion of consistency that is *local* to the FSP and the user's data must be ensured. Specifically a local notion of consistency requires the current privacy policy of the FSP not to violate the data owners privacy preferences with respect to the FSP most up-to-date version of the user's data and privacy preferences. Global consistency can be derived by extending the notion of local consistency to all FSPs in a FIM and impose proper updates at each change.

Inconsistency in case of update of users' privacy preferences is handled as described by Algorithm 5. This type of inconsistency can be detected at time T either by a FSP or by a user. Note that different time points may correspond to different policy versions and data updates, so an inconsistency found at time T may be short lived. That is, the inconsistency may not exist at time $T - \epsilon$, where ϵ is the minimum amount of time that it takes for a policy, preference, or data value to get updated at the particular FSP.

Algorithm 5 is organized into two main phases: first is the update (lines 1-7), followed by the actual tracing (8-17). The *update* phase begins with the user (or FSP on behalf of him) running the tracing algorithm, presented in Appendix A to identify FSPs having data which may require an update as a result of the new privacy preferences. Inconsistency can be detected at any time T_z subsequent to T_{update}, where T_{update} corresponds to the most recent update. Finding an inconsistency implies a fault if and only if the updated *PPref* changed the matching result (line 9) and this has not been reflected neither in a corresponding *PPol* change nor in a notification to the data owner. Instead, inconsistency is not faulty if the FSP was not notified of the change and/or there is no mismatch according to the privacy preference version used by FSP. Additionally, lack of propagation may be due to the specific strategy adopted in the tracing, which may not be exhaustive. For example, the user may have employed a hybrid strategy comprising of a timed priority combination wherein the timer was set to some small number t and the priority for FSPs in the *.edu* domain. Thus the user may not have obtained all the FSPs (especially those outside of the *.edu* domain), and therefore not performed a comprehensive update. If the service FSP_m should not be a data holder and is detected during the trace, then this not only raises the problem of inconsistency but is also indicative of a privacy breach.

The case of data changes requires users' new data's version to be propagated as well, and it can be easily achieved with the help of ours strategies An FSP detect data inconsistency when it obtains multiple versions of the same data from different sources. This can occur, for example, if FSP_3 receives two values for d, d' and d'', respectively from FSP_2 and FSP_1, where $d' \neq d''$. Inconsistencies of this kind can be *strong* or *weak*. Briefly, *strong* inconsistency holds when the inconsistency hinders the FSP from using the data. *Weak* inconsistency relates to cases that do not affect FSP's intended use of d. To classify an inconsistency into weak or strong, several approaches are possible, such as editing distance [26] or semantic matching [13].Alignment scores, currently used commonly for biomedical sequences of letters, and *degree* of inconsistencies are other examples of possible difference measure.

Even if the $FSPs$ are using inconsistent data, the FSP should only be concerned in case that inconsistency is strong. If this type of inconsistency is detected, the FSP proceeds by picking either d or d' based on local criteria, such as the most trusted source, between FSP_1 and FSP_2. Alternatively, if inconsistency is strong and not

Algorithm 1 Consistency Protocol

Require: Input parameters: Current time cT, $PPref$ privacy preferences of data d at time cT is $PPref^d_{cT}$, and receiver FSP is denoted by FSP_{rec}
Ensure: Detect fault inconsistency
1: **Data Update** {Traverse FIM and update each FSP}
2: Update(u, **PPref**) {Update u's preferences at current FSP}
3: $FSPList = GM\_TRACE(T, d, f)$ {Get list of FSP's that have d at time cT}
4: $T_{update} = cT$
5: **for all** $FSP \in FSPList$ **do**
6: Update(u, **PPref**)
7: **end for**
8: **Inconsistency detection**
{ At time $T_{update+\Delta}$ an inconsistency is found at FSP_m for d - there are no updates between T_{update} and $T_{update+\Delta}$}
9: **if** $\mathbf{PPol^d}$ $\neg\mathcal{M}$ $\mathbf{PPref^d}$ **then**
10: **if** $FSP_m \in FSPList$ **then**
11: Faulty inconsistency
12: **else if** $n.Ts' \geq T_{update}$ **then**
13: Faulty inconsistency
14: **else**
15: Update(u, **PPref**) {Not faulty inconsistency, due to lack of propagation}
16: **end if**
17: **end if**

solvable, the FSP contacts the user for clarification of which data to keep and notify him/her that different versions of his data exist, so that the user can take run the update procedure.

6. RELATED WORK

Our work is motivated from the existing initiatives related to federated digital identity management. In the corporate world there are several emerging standards for identity federations, such as the ones compliant to Liberty Alliance standards [11] and WS-* standards [24]. The goal of such frameworks is to provide a controlled environment for managing identities of federated users. However, none of them provide policy subsumption/checking capabilities: they primarily rely on contractual agreements and on simple mechanisms to check for user's privacy preference (similarly to P3P), at the time of disclosing data. Additionally, they do not provide any mechanism to enable data tracing distributed across the federation. Our approach can be leveraged to provide additional "policy compliance checking and monitoring" capabilities, complementary to these existing mechanisms. Similar considerations apply to other federated identity management frameworks, such as OpenId [18], Higgins [7] and Shibboleth [10].

Regarding policy specification, the Platform for Privacy Preferences Project (P3P) is a standardized platform that provide an XML based policy specification language, to be used to specify an organization's privacy practices in a way that can be parsed and used by policy checking agents on the user's behalf [6]. Many user software agents are currently available for use, which handle policy checking and invoke the required actions that need to occur when a website's policy is found to conflict with the user's preferences. Such tools could be useful in our system when the user matches his preference with the FSP's privacy policies. Other works closely related to ours include the privacy platforms proposed for enterprises that separates the enterprise-specific deployment policies from the privacy policies that cover the complete life cycle of collected data. Two such approaches are E-P3P [15] and IBM's Enterprise Privacy Authorization Language (EPAL)[8]. Also, the enhanced SAML [21] can be considered as a way to support user friendly privacy/preference expressions. E-P3P [15] provides for privacy-enabled management and exchange of customer data. E-P3P and EPAL, among other features, provide a privacy policy specification language specifically designed for organizations to specify their internal privacy policies. In our work, we focus on different issues from the ones tackled by such projects, in that we do not deal with policy representation but propose approaches to check organization's compliance with users' privacy preferences based on policy traceability. Tivoli [9] is the commercial version of E-P3P, is a privacy protection framework that extends traditional access control systems by adapting EPAL privacy oriented language.

In [20, 16, 2] and [17] privacy preserving frameworks for federated systems were proposed. PREP [1] is another privacy preference expression language used by the Liberty enabled attribute providers to collect and store the user's privacy preferences. It shares the same extension formats with APPEL [**?**] but is more restrictive than APPEL. It also supports the multi-level privacy policy approach in Liberty Alliance specifications. It facilitates the decision process in legitimately releasing attributes at the attribute provider by comparing the policy level in the request with the level in PREP at the attribute provider's site.

In [20] a first tracing mechanism was proposed. Our current work is superior to [20] in that we propose multiple strategies to conduct tracing and we tackle data and privacy preference consistency issues, that were not discussed in previous work. Another recent relevant work which is extremely relevant for our research is [25]. The authors start from similar consideration than ours, i.e., the need of solutions that protect user's privacy without relying on cryptographic and restrictive technical solutions, to motivate their proposed framework, which is based on users' accountability. They suggest that accountability must become a primary means by which society addresses issues of appropriate use, and propose to extend the Web architecture to support transparency and accountability. Our approach entails a form of accountability since an entity non-compliant with the users original privacy preferences can be identified.

Finally, an interesting piece of work is from Gross[5] and colleagues, who proposed a schema to incorporate anonymous credentials in FIM systems. Such an approach achieves users' privacy by guaranteing anonymity, but it requires organizations to be compliant on the users' attribute representation used. Additionally, anonymity is not applicable in several domains where users identity is required, and alternative privacy preserving mechanisms are needed.

7. CONCLUSION

In this paper we address the problem of privacy in a federated environment by satisfying two important requirements. The first requirement is to provide mechanisms

making it possible for users to trace their PII across the federation, and verify whether their data has been managed according to their privacy preferences. The second requirement is to analyze inconsistency issues in the FIM and provide an algorithmic approach to detect relevant inconsistency cases. Our approach entails a form of accountability since an entity non-compliant with the users original privacy preferences can be identified. Future work includes a formal analysis of the inconsistency cases, in particular with respect to data inconsistency. The solution proposed in this paper can be easily applied in case of remotely distributed organizations, where multiple organizations share users' data across geographical boundaries. It would be interesting to explore how possibly different heterogeneous regulatory systems impact our tracing based solution.
An implementation of the main protocols presented in the paper is currently ongoing.

8. REFERENCES

[1] Gail-Joon Ahn and John Lam. Managing Privacy Preferences for Federated Identity Management. In *Digital Identity Management*, pages 28–36, 2005.

[2] Gail-Joon Ahn and John Lam. Managing Privacy preferences for federated identity management. In *Digital Identity Management*, pages 28–36, 2005.

[3] Franz Baader and Bernhard Hollunder. Embedding defaults into terminological knowledge representation formalisms. *J. Autom. Reasoning*, 14(1):149–180, 1995.

[4] Abhilasha Bhargav-Spantzel, Jan Camenisch, Thomas Gross, and Dieter Sommer. User centricity: a taxonomy and open issues. *To appear in Journal of Computer Security.*

[5] Jan Camenisch, Thomas Gross, and Dieter Sommer. Enhancing privacy in identity federation anonymous credentials ensure unlinkability in ws-security. In *IEEE Workshop on Web Services Security*, 2006.

[6] Lorrie Cranor, Marc Langheinrich, Massimo Marchiori, and J. Reagle.

[7] Higgings. Open Source Initiative, http://www.eclipse.org/higgins, 2007.

[8] http://www.zurich.ibm.com/security/enterprise privacy/epal/. EPAL 1.0 Specification.

[9] IBM Tivoli Software. http://www-306.ibm.com/software/tivoli/, 2007.

[10] Internet2. Shibboleth. http://shibboleth.internet2.edu.

[11] Liberty Alliance Project. http://www.projectliberty.org.

[12] Security Breaches at UCLA Medical Center More Privacy. http://www.privacy.org/archives/002241.html (accessed may 2008).

[13] N. Noy and M. Musen. Using non-local context for semantic matching. In *IJCAI 2001 workshop on ontology and information sharing, Seattle (WA US)*, pages 63–70, 2001.

[14] P3P Preference Exchange Language 1.0 (APPEL1.0). http://www.w3.org/tr/p3p-preferences/.

[15] Gurieth Karjoth Paul Ashley, Satoshi Hada and Matthias Schunter. E-P3P Privacy Policies and Privacy Authorization. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES)*, 2001.

[16] Samir Saklikar and Subir Saha. User privacy-preserving identity data dependencies. In *DIM '06: Proceedings of the second ACM workshop on Digital identity management*, pages 45–54, New York, NY, USA, 2006. ACM.

[17] Farzad Salim, Nicholas Paul Sheppard, and Reihaneh Safavi-Naini. Enforcing P3P policies using a digital rights management system. In *Privacy Enhancing Technologies*, pages 200–217, 2007.

[18] SourceID: Open Source Federated Identity Management. http://www.sourceid.org/resources/basics.html.

[19] Anna C. Squicciarini, Marco Casassa Mont, Abhilasha Bhargav-Spantzel, and Elisa Bertino. Automatic compliance verification of privacy policies in federated digital identity management. 2008.

[20] Anna Cinzia Squicciarini, Abhilasha Bhargav-Spantzel, Alexei Czeskis, and Elisa Bertino. Traceable and automatic compliance of privacy policies in federated digital identity management. In *Privacy Enhancing Technologies*, pages 78–98, 2006.

[21] OASIS Standard. Security assertion markup language (saml) v2.0, 2005.

[22] Philip Sutton and Charles D. Hansen. Isosurface extraction in time-varying fields using a temporal branch-on-need tree (t-bon). In *VIS '99: Proceedings of the conference on Visualization '99*, pages 147–153, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.

[23] Cheryl Vroom and Rossouw von Solms. Towards information security behavioural compliance. *Computers and Security*, 23(3):191–198, May 2004.

[24] Web Services Standard Listings. http://en.wikipedia.org/wiki/ list_of_web_service_specifications, 2007.

[25] Daniel J. Weitzner, Harold Abelson, Tim Berners-Lee, Joan Feigenbaum, James Hendler, and Gerald Jay Sussman. Information accountability. *Commun. ACM*, 51(6):82–87, 2008.

[26] Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262, 1989.

APPENDIX

Algorithm 2 Hybrid Resolution Algorithm

Require: Input resolution parameters: Current time cT, Data d, privacy preferences of d at time cT is $PPref^d_{cT}$, and receiver FSP is denoted by FSP_{rec}
Ensure: Source input validity
1: $PPol^d_{FSP_{rec}:cT} := getPolicy(FSP_{rec}, d, cT)$ { Get Privacy Policy of FSP_{rec}}
2: $TimeAlloted := GetTraceTimeLength()$
3: $TimeStarted := cT$
4: HYBRID TRACE($TimeAllotted, FSP_{rec}, d$);
5: **return**
{The main hybrid function:}
6: Begin function HYBRID TRACE ($TimeAllotted, FSP_{rec}, d$)
7: list $FSP_{dests} := getDestFSP(FSP_{rec}, d)$ {Get list of FSPs that shared data d }
8: **if** $FSP_{dests} = NULL$ **then**
9: **Return** End of trace reached
10: **end if**
11: $TimeRemaining = TimeAlloted - (TimeStarted - cT)$
12: **if** $TimeRemaining < 0$ **then**
13: **Return** End of trace reached
14: **end if**
15: **for** each $FSP_i \in FSP_{dests}$ **do**
16: $PriorityFSP_i = GetPriority(FSP_i)$
17: HYBRID TRACE($TimeRemaining \times PriorityFSP_i, FSP_i, d$)
18: $PPol^d_{FSP_{dest}:cT} := getPolicy(FSP_{dest}, d, cT)$
19: $PPol^d_{FSP_{rec}:cT} := getPolicy(FSP_{rec}, d, cT)$
20: **if** NOT($subsume(PPol^d_{FSP_{rec}:cT}, PPol^d_{FSP_{dest}:cT})$) **then**
21: $T = getSharedTime(d, FSP_{rec}, FSP_{dest})$
22: **if** ($subsume(PPol^d_{FSP_{rec}:T}, PPol^d_{FSP_{dest}:T})$) **then**
23: PRINT "Incompatible with current preferences, compatible with time T settings"
24: **else**
25: **Return** Mismatch found with FSP_{rec} and FSP_{dest}
26: **end if**
27: **end if**
28: list $FSP_{dests} := getDestFSP(FSP_{dest}, d)$
29: $FSP_{rec} := FSP_{dest}$
30: **end for**
31: **Return** End of trace reached

Algorithm 3 Priority Based Resolution Algorithm

Require: Input resolution parameters: Current time cT, Data d, privacy preferences of d at time cT is $PPref^d_{cT}$, and receiver FSP is denoted by FSP_{rec}
Ensure: Source input validity
1: $PPol^d_{FSP_{rec}:cT} := getPolicy(FSP_{rec}, d, cT)$ { Get Privacy Policy of FSP_{rec}}
2: list $FSP_{dests} := getDestFSP(FSP_{rec}, d)$ {Get list of FSPs that shared data d }
3: **if** $FSP_{dests} = NULL$ **then**
4: **Return** End of trace reached
5: **end if**
6: $FSP_{dest} := HighestPriority(FSP_{dests})$
7: **if** $FSP_{dest} = NULL$ **then**
8: **Return** PRIORITY ERROR
9: **end if**
10: $PPol^d_{FSP_{dest}:cT} := getPolicy(FSP_{dest}, d, cT)$
11: $PPol^d_{FSP_{rec}:cT} := getPolicy(FSP_{rec}, d, cT)$
12: **if** NOT($subsume(PPol^d_{FSP_{rec}:cT}, PPol^d_{FSP_{dest}:cT})$) **then**
13: $T = getSharedTime(d, FSP_{rec}, FSP_{dest})$
14: **if** ($subsume(PPol^d_{FSP_{rec}:T}, PPol^d_{FSP_{dest}:T})$) **then**
15: PRINT "Incompatible with current preferences, compatible with original time T settings"
16: **else**
17: **Return** Mismatch found with FSP_{rec} and FSP_{dest}
18: **end if**
19: **end if**
20: $flagFSP - passed(FSP_{dest})$
21: list $FSP_{dests} := getDestFSP(FSP_{dest}, d)$
22: $FSP_{rec} := FSP_{dest}$ {Going to next hop of the trace}
23: GOTO Step 3