Relationship-Based Access Control for Resharing in Decentralized Online Social Networks

Richard Gay[1], Jinwei Hu[1], Heiko Mantel[1], and Sogol Mazaheri[2]

[1] Modeling and Analysis of Information Systems,
Department of Computer Science, TU Darmstadt, Germany
lastname@mais.informatik.tu-darmstadt.de
[2] Cryptography and Complexity Theory,
Department of Computer Science, TU Darmstadt, Germany
sogol.mazaheri@cryptoplexity.de

Abstract Decentralized online social networks (DOSNs) have adopted quite coarse-grained policies for sharing messages with friends of friends (i.e., resharing). They either forbid it completely or allow resharing of messages only without any possibility to constrain their subsequent distribution. In this article, we present a novel enforcement mechanism for securing resharing in DOSNs by relationship-based access control and user-determined privacy policies. Our mechanism supports resharing and offers users control over their messages after resharing. Moreover, it addresses the fact that DOSNs are run by multiple providers and honors users' choices of which providers they trust. We clarify how our mechanism can be effectively implemented by a prototype for the DOSN Diaspora*. Our experimental evaluation shows that controlling privacy with our prototype causes only a rather small performance overhead.

Keywords: decentralized online social networks, privacy, access control

1 Introduction

Online social networks (OSNs) are web-based services that offer users the functionality to share messages with other users. A decentralized online social network (DOSN) [12] is an OSN that is supported by multiple service providers. In a DOSN, a user can choose a provider whom she trusts most to store her profile. Typical OSNs provide an author with means for sharing a message with the set of users she categorized as 'friends', 'colleagues', etc. As of today, DOSNs allow authors to share sensitive messages with selectable sets of users but forbid resharing of sensitive messages entirely.[3]

A better support of controlled resharing in DOSNs would be beneficial. Consider, for instance, a user who visits various US national parks. Before the trip, she had informed her friends early that she will visit the US and later

[3] Even the centralized OSN Facebook supports controlled resharing only with users with whom the message had been already shared with. The alternative in Facebook is uncontrolled sharing where users may arbitrarily reshare messages that they receive.

informed them about the exact route and the dates of her visits. During the trip, she enjoys the landscape and sharing pictures with her friends. For privacy reasons, she wants to control spreading of this information: pictures should remain among her direct friends and dates of her visit among her friends and her friends' closest friends. Her motivation for limiting spreading of the dates of her trip could be to not provoke burglary [25]. She is less concerned about distributing the mere fact that she is visiting US national parks. This information may be distributed further, but without becoming public. This scenario illustrates the need for providing fine-grained control over sharing, resharing, and the distribution of reshared messages. We refer to the combination of these three forms of controlled information dissemination by the term *controlled resharing*.

In this article, we propose a privacy enforcement mechanism for controlled resharing in DOSNs. Our mechanism enables users to specify by privacy policies to which extent messages that they are sharing with others may be distributed further. Our mechanism provides control over the dissemination of messages inside a DOSN, ensuring that the privacy policies of all users are respected.

Conceptually, our enforcement of privacy policies is based on *relationship-based access control (ReBAC)* [15,13]. When checking authorization, we take the relationships of all users into account who were involved in delivering a message to the user who wishes to distribute this message further. Technically, we capture the relationship between users by trust values, where each user can define her personal trust values for categories of users. The decision whether a received message may be distributed to some category of users is made based on the concept of *trust concatenation* [19].

We developed an implementation of our ReBAC mechanism[4] for Diaspora* [18], at the time of writing the most popular DOSN [28]. To accommodate the distributed nature of DOSNs, our mechanism also has a distributed architecture. We chose a design that supports making authorization decisions in a decentralized fashion to avoid a single point of failure and performance bottlenecks. As underlying technological platform, we chose the CliSeAu tool [16]. This combination of design decisions results in a solution for enforcing controlled resharing in Diaspora* that is both, effective and efficient. In our performance evaluation, we observed an overhead of less than 2% for resharing in the domain of the same provider and of less than 4% when controlling resharing across providers.

2 Definition of Privacy Policies

After the author of a message m has shared m with a collection of categories (such as 'friends', 'family', or 'colleagues'), some users in these categories might reshare m with others, some recipients of such a reshared message might reshare m again, and so on. To capture how a message has been delivered from its author u_1 to a set C_n of categories, we use lists of the form $(u_1, C_1, \ldots, u_n, C_n)$, which we call *reshare paths*.

[4] Available at `http://www.mais.informatik.tu-darmstadt.de/CReDiC.html`.

For instance, the reshare path $(\mathit{Alice}, \{\mathit{Colleagues}, \mathit{Friends}\}, \mathit{Bob}, \{\mathit{Family}\})$ captures the sharing of a message by Alice with her friends and colleagues and the subsequent resharing by recipient Bob with his family.

When resharing a message, the author risks that recipients might abuse sensitive content. Hence, the distribution of sensitive messages needs to be limited to receivers whom the author sufficiently trusts. The role of trust in making the decision to share or not to share a message is well captured by the notion of *decision trust*, "the extent to which one party is willing to depend on something or somebody in a given situation with a feeling of relative security, even though negative consequences are possible" [22].

We model the trust of a user u in her category c by a scalar *trust value* from the interval $[0, 1]$, where greater values mean greater trust. The maximal trust value 1 means that u trusts users in c as much as herself wrt. the propagation of her messages. The minimal trust value 0 means that u does not have any trust in users in c wrt. propagation. We capture the trust values of a user, along with the user's categories and relationships to other users in the user's privacy policy:

Definition 1. *A* privacy policy of a user u *is a triple* $pp_u = (\mathrm{CAT}_u, rel_u, tv_u)$, *where* CAT_u *is a set,* $rel_u \subseteq \mathrm{CAT}_u \times \mathrm{USER}$ *is a binary relation and* $tv_u \colon \mathrm{CAT}_u \to [0, 1]$ *is a function. A* privacy policy for a set of users $U \subseteq \mathrm{USER}$ *is a family* $(pp_u)_{u \in U}$ *of privacy policies for each user in* U.

In a privacy policy, CAT_u specifies all categories of u. The relation rel_u captures which other users are in the categories of user u. For instance, $rel_{Alice}(\mathit{Friends}, \mathit{Bob})$ captures that Bob is a member of Alice's $\mathit{Friends}$ category. The function tv_u captures the trust of user u in her categories. We impose no further constraints on privacy policies. Hence, through $rel_{Alice}(\mathit{Colleagues}, \mathit{Alice})$ and $tv_{Alice}(\mathit{Colleagues}) = 0.5$, Alice could specify a medium trust in herself as a colleague.

Intuitively, rel_u specifies which users from the universe USER of all users may receive a message that u (re)shares with a particular category. That is, rel_u captures an expectation of u about the visibility of messages that she shares with her categories. The function tv_u captures a complementary aspect, namely to which extent u trusts users in her categories to propagate her sensitive messages.

Sensitivity of messages is not part of privacy profiles. We capture the sensitivity of messages for authors by values in $[0, 1]$ (greater values mean greater sensitivity).

We denote the trust of a message's author in a recipient u', who obtained a message m via a reshare path π, under a privacy policy $(pp_u)_{u \in U}$ for a set U comprising all users in π by $\mathcal{PT}((pp_u)_{u \in U}, \pi, u')$. We define $\mathcal{PT}((pp_u)_{u \in U}, \pi, u')$ recursively over the length of the reshare path:

$$\mathcal{PT}((pp_u)_{u \in U}, (u, C), u') = \max(\{tv_u(c) \mid c \in C \wedge rel_u(c, u')\} \cup \{0\})$$
$$\mathcal{PT}((pp_u)_{u \in U}, \pi.(u, C), u') = \mathcal{PT}((pp_u)_{u \in U}, \pi, u) \cdot \mathcal{PT}((pp_u)_{u \in U}, (u, C), u')$$

That is, if a user u' received a message m from m's author u directly, then the trust of u in u' via (u, C) equals u's maximal trust value for a category of u from C that u' is in. If there is no such category, then the trust of u in u' is 0. If u' received m via a longer path π then the trust of m's author in u' via π

is the product of the trust values for each (re)sharing of m by a user along π. We discuss the choice in Section 6. The product of trust values ensures that prolonged paths yield a decreased trust value.

We make the semantics of privacy policies and their impact on sharing and resharing of messages more precise by the following definitions. As a prerequisite for the definitions, we say that a reshare path (u, C) (resp. $\pi.(u, C)$) is a *reshare path to user* u' iff $rel_u(c, u')$ holds for some category $c \in C$.

Definition 2. Sharing *of a message* m *with sensitivity value* $s \in [0, 1]$ *by a user* u *with a set of categories* C complies with a privacy policy $(pp_u)_{u \in U}$ *if and only if* $s \neq 1$ *holds,* $C \subseteq \mathrm{CAT}_u$ *holds, and* (u, C) *is a reshare path to* u' *for all users* u' *who receive* m *due to this sharing.*

Definition 3. *Let* $sc \in [0, 1]$ *be arbitrary.* Resharing *of a message* m *with sensitivity value* $s \in [0, 1]$ *which had been received via a reshare path* π*, by a user* u *with a set* C *of categories* complies with a privacy policy $(pp_u)_{u \in U}$ *if and only if* $s \neq 1$ *holds;* $C \subseteq \mathrm{CAT}_u$ *holds;* (u, C) *is a reshare path to* u' *for all users* u' *who receive* m *due to this sharing; and* $\mathcal{PT}((pp_u)_{u \in U}, \pi, u) \geq \frac{sc}{1-s}$.

The inequality condition introduced in Definition 3 establishes a lower bound on the trust for resharing ($\frac{sc}{1-s}$) that the author of a message can raise through an increased sensitivity value. Note that Definition 3 is parametric in $sc \in [0, 1]$, where higher values of sc are more restrictive. We refer to this parameter as *sensitivity coefficient*. For instance, the sensitivity coefficient $sc = 0.35$ ensures that resharing is completely forbidden along reshare paths π with low trust ($\mathcal{PT} < 0.3$), is allowed for low-sensitivity messages ($s < 0.3$) along reshare paths π with medium trust ($0.5 \leq \mathcal{PT} \leq 0.6$), and is completely forbidden for messages with a sensitivity value above 0.65. Sensitivity coefficients have been used before in the semantics of privacy policies for controlling the direct sharing of messages between users. For instance, Kumari et al. [24] propose sensitivity coefficients for different kinds of operations. Definition 3 transfers this concept to resharing.

Our privacy policies augment what can typically be found in OSNs, namely categories of users, by trust and sensitivity. Based on these ingredients in privacy policies, we define when sharing and resharing are compliant. Since the presented semantics leaves underspecified how users' privacy policies are obtained for checking compliance, the semantics supports a decentralized storage of policies as well as dynamically changing policies.

3 The ReBAC Mechanism

We chose a distributed architecture for our mechanism to accommodate the distributed nature of DOSNs. Each service provider of a protected DOSN is supervised by a separate controller. The controller at a provider ensures that all sharing/resharing actions of users whose profiles the provider stores comply with the privacy policies of all users, not only of users hosted at the provider.

In the design of our ReBAC mechanism, we assume that a user's messages are only distributed to service providers who are controlled by our mechanism and

who can be trusted to not circumvent our control. This could be achieved, e.g., by corresponding legal agreements between service providers: Service providers who did not sign the agreement are excluded from receiving certain messages by providers who signed the agreement.

Our mechanism provides no protection against communication outside the DOSN. For instance, a user with whom a message has been shared or reshared inside the DOSN might take this message and communicate it to others via email or might use the browser to copy the message text and paste it, possibly with modifications, into a new message in the DOSN. This risk cannot be completely mitigated by technical means. Users should take this aspect into account when specifying trust values and when admitting users to their categories. A user who is distrusted should not be given a sensitive message in the first place.

Finally, we assume that the implementation of sharing/resharing with categories in the DOSN is sound in the sense that when a user shares or reshares a message m with a category c then only users in c receive m.

3.1 Decentralized Control of Resharing

The purpose of our ReBAC mechanism is to ensure that privacy policies of all users are obeyed when sharing and resharing messages inside a DOSN. Since we assume the DOSN to soundly implement sharing and resharing of messages with categories, compliance of sharing is ensured by the DOSN (recall Definition 2). Our controllers therefore do not control the sharing of a message by a user. Compliance of resharing, however is only partially ensured by the DOSN, leaving one crucial condition to be ensured by our controllers (recall Definition 3). When a user u who obtained a message m via a reshare path π attempts to reshare m with a set C of categories, our controller therefore checks:

$$\boxed{\text{Does } \mathcal{PT}((pp_u)_{u\in U}, \pi, u) \geq \tfrac{sc}{1-s} \text{ hold?}} \quad (\mathsf{C})$$

Intuitively, the check (C) ensures that the trust of m's author into u is sufficiently high to reshare m with other users. The check involves the privacy policy of (at least) all users in π, which are part of the users' profiles and, hence, possibly stored at another service provider than u's. For enabling the controllers to perform the check, our controllers establish the availability of all relevant information by *coordination* as follows.

sharing m of sensitivity s by u at provider sp with categories C:
When this action is performed, the controller at sp disseminates the following information to the controller of each service provider sp' with a recipient of m: the initial reshare path, the sensitivity s, and for each recipient u' the trust value $\mathcal{PT}(pp_u, (u, C), u')$. Figure 1 (left) shows the dissemination procedure in pseudo-code. For the sharing, it is invoked with the empty path π and the value $pt = 1$ representing the trust value for the empty path. Each recipient stores the values for controlling future reshares.

resharing m by user u at provider sp with categories C:
When this action is performed, the controller checks whether the action complies with users' privacy policies by performing check (C). If the check

```
disseminate(m, π, u, C, s, pt):
  Data: sp, pp_u = (CAT_u, rel_u, tv_u)
  U ← {u' | ∃c ∈ C : rel_u(c, u')}
  for sp' ∈ {sp(u') | u' ∈ U}\{sp} do
    PT ← ∅
    for u' ∈ {u' ∈ U | sp(u')=sp'} do
      pt' ← pt · PT(pp_u, (u, C), u')
      PT ← PT ∪ {(u', pt')}
    end
    send(con@sp', (m, π.(u, C), s, PT))
  end
```

```
Data: sp, u, m, C, (pp_u)_sp(u)=sp, sc
s ← sensitivity value for m
π ← reshare path for m to u
decompose π = π1.π2, where π2 is
   maximal with only users from sp
pt1 ← trust value for π1 to first user in π2
pt ← pt1 · PT((pp_u)_sp(u)=sp, π2, u')
if pt ≥ sc/(1−s) then
  disseminate(m, π, u, C, s, pt)
else
  disallow reshare
```

Figure 1. Algorithms for decentralized coordination among controllers

succeeds, then the controller disseminates the reshare path, sensitivity, and trust values to all affected service providers, as in the case of sharing. Otherwise, the controller disallows the resharing. Figure 1 (right) shows the procedure in pseudo-code. For the check, the controller uses its local privacy policies $(\mathsf{pp}_u)_{\mathtt{sp}(u)=sp}$ as well as the value pt_1 obtained when m was delivered to the controller's service provider. If m was never received from another service provider (i.e., π_1 is empty), then $\mathsf{pt}_1 = 1$.

The coordination among controllers follows the propagation of messages. When a sharing/resharing causes a message to be delivered to another service provider, the information exchanged by the controllers enables the receiving controllers to perform check (C) for future reshares. No further communication among the controllers is then required for this check. That is, all coordination is decentralized.

3.2 Decentralized Control with Timely Policies

The approach presented in Section 3.1 has the virtue to require no coordination among controllers for checking whether a reshare complies with users' privacy policies. This virtue comes with a drawback, which we address in this section: When a controller checks whether a reshare of a message obtained from another service provider is compliant, it might rely on outdated privacy policies underlying the pt-value for π_1 in Figure 1. That is, once a message has been shared/reshared with users at another provider, changes of privacy policies by users at the author's service provider cannot influence the further propagation of the message anymore.

We propose a decentralized approach for checking whether a reshare complies with timely privacy policies of users. The key idea behind our approach is to have the controllers perform a decentralized computation of the check (C) and to refrain from any proactive distribution of privacy profile information. For the decentralized computation, we transform the check (C) slightly to:

$$\boxed{\text{Does } (1-\mathsf{s})\cdot\mathcal{PT}((\mathsf{pp}_u)_{u\in U},\pi,\mathsf{u}) \geq \mathsf{sc} \text{ hold?}} \quad (\mathsf{C}')$$

The left-hand side of (C') contains all profile information (the privacy policies and the sensitivity value) and is the subject to the decentralized computation.

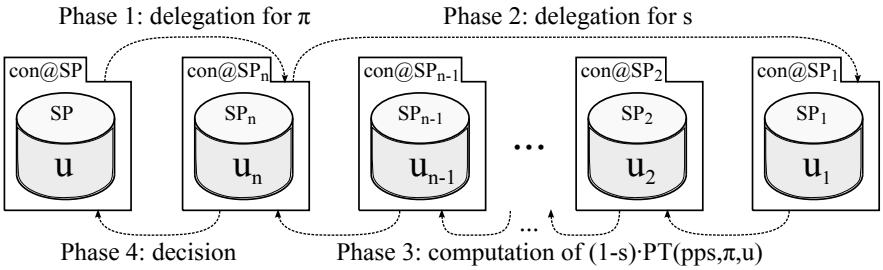

Figure 2. Coordination for decentralized control with timely policies

Notably, in general none of s, $(pp_u)_{u \in U}$, and π is known to the controller performing the check. Suppose a user u wants to reshare a message m. The decentralized computation proceeds in four phases:

Phase 1: The controller at u's service provider queries the user u' from which u received m. The controller then delegates check (C') to the controller at the service provider of u'. The latter controller queries the reshare path $\pi = (u_1, C_1, \ldots, u_n, C_n)$ of m to $u' = u_n$.

Phase 2: The controller for u_n delegates the check to the controller for u_1, along with π. The controller for u_1 queries the sensitivity value s for m and initializes result R of the decentralized computation by assigning $R \leftarrow 1-s$.

Phase 3: The active controller (initially the controller for u_1) takes the longest prefix π_1 of π such that all users in π_1 have their profiles at the service provider of the controller. The controller then updates the result by assigning $R \leftarrow R \cdot \mathcal{PT}((pp_u)_{u \in U}, \pi_1, u_2)$, where U contains all users in π_1 and where u_2 is the first user in the remaining suffix π_2 of π. If π_2 is non-empty, the controller delegates the further computation, along with R for the result and π_2 for the reshare path, to the controller of u_2, which then proceeds in Phase 3. If π_2 is empty, then Phase 4 is entered.

Phase 4: The active controller (for u_n) checks whether $R \geq sc$ holds. Depending on the result of the check, the controller sends the decision to allow or to disallow the reshare to the controller of u. The controller for u implements this decision and records π for future reshares.

In Phase 1, we exploit that the user u' who (re)shared message m is part of m. Moreover, the controller at u_n knows π for m, as established inductively over the length of the reshare path in Phase 4. In Phase 2, we exploit that the sensitivity value s for m is stored at the service provider of u_1.

Figure 2 visualizes the four phases for the case that all users on the reshare path are at different service providers (SP_i), each of which having its individual controller ($\mathrm{con@SP}_i$). The coordination is triggered by the controller at u's service provider. The four phases then sequentially activate the remaining controllers in the ordering indicated by the arrows.

The computation we propose avoids to gather users' privacy policies at a central location. Only intermediate computation results are provided to the

involved controllers and are subsequently discarded again. The computation leads the mechanism to effectively check (C) in a decentralized fashion based on timely privacy policies of users. The coordination among controllers in the computation follows the ordering of service providers in π, which ensures that successively involved providers are bound by contract.

3.3 Optimized Coordination

We propose two optimizations for reducing the amount of coordination in the decentralized computation presented in Section 3.2.

The first optimization applies when a service provider occurs more than once in a reshare path but other service providers occur in between. The optimization augments Phase 1, in which the controller for u_n additionally computes the set SP of all service providers in π and passes this set to the subsequent phases. Phase 3 is replaced by the following:

Phase 3': The controller computes $R' = R \cdot \prod_{u_i \in U} \mathcal{PT}(pp_{u_i}, (u_i, C_i), u_{i+1})$, where U is the set of all users whose profile is stored at the service provider of the controller and where $u_{n+1} = u$. The controller then removes itself from SP. If the resulting set is empty, then Phase 4 is entered. Otherwise, the controller delegates the further computation, along with R' for the result, the updated SP, and with the unchanged reshare path π, to some controller in SP who then proceeds in Phase 3'.

In Phase 3', each controller is activated at most once. The optimization is sound and precise due to the associativity and commutativity of multiplication. However, it does in general not preserve that service providers of successively involved controllers are bound by contract.

The second optimization particularly affects long reshare paths and reshare paths containing low trust values. It augments Phase 3 by the additional abort condition $\boxed{\text{Is } R' < sc?}$ that triggers the transition to Phase 4. With this condition, the computation terminates once a sufficiently low intermediate result R is encountered. The optimization is sound and precise because the product in the definition of function $\mathcal{PT}$ is monotonically decreasing in further factors, as each of the factors equals a trust value $tv_u(c) \in [0, 1]$. Both optimizations can soundly and precisely be combined, and each maintains a decentralized computation.

4 A Prototype for Diaspora*

To demonstrate the feasibility of our ReBAC mechanism, we developed *CReDiC*, short for "Controlled Resharing in Diaspora* with CliSeAu." CReDiC implements the mechanism for Diaspora*, the popular open-source DOSN. The implementation utilizes timely privacy policies (as described in Section 3.2) with optimized coordination (as described in Section 3.3). Diaspora* is a suitable candidate for CReDiC as its sharing and resharing with categories is sound. As the underlying technological platform of CReDiC, we utilize CliSeAu [16].

4.1 CliSeAu for Ruby

CliSeAu is a tool for dynamic policy enforcement in distributed programs [16]. Previously, it supported enforcement for Java programs only. We developed an extended variant of CliSeAu that supports enforcement for Ruby programs in addition. This was necessary for building CReDiC on top of CliSeAu, as Diaspora* is implemented in Ruby. Our extension utilizes Aquarium [30] for instrumenting Ruby programs. It consists of 244 lines of Java code and 38 lines of Ruby code.

Our extension of CliSeAu retains the high-level architecture and the coordination model used by CliSeAu for the mechanisms it generates. That is, the mechanisms consist of enforcement capsules (ECs), placed at the individual components of the distributed target program. At runtime, each EC intercepts policy-relevant events of one component of the target, makes decisions for intercepted events, and enforces the decisions made. The developer of an enforcement mechanism using CliSeAu can specify the events to intercept, the decision-making, the enforcement, and the coordination among multiple ECs.

4.2 Mapping Diaspora* on our Trust Model

We instantiate the trust model introduced in Section 2 for Diaspora* as follows. Diaspora* supports that users organize their acquaintances into categories (called "aspects" in Diaspora*) and that users can change the set of their categories from the default categories provided by Diaspora*. A user's set of categories corresponds to the set CAT_u in our model. An acquaintance is either in a category of a user or not, which is captured by the relations rel_u in our model. When a user wants to share/reshare a message, she can select one or multiple of her categories to share with. This corresponds to how we model sharing and resharing with sets of categories.

Trust and sensitivity values are not supported by Diaspora*. We augment Diaspora* by trust values for categories by suffixing the category names with their trust value – e.g., "family (0.9)". Our mechanism separates name and trust value again to allow users to change trust by renaming the category. We simulate the sensitivity value of a message by utilizing the least trust value among the categories with which the message is shared.

Diaspora* prohibits resharing of sensitive messages, i.e., of messages not classified as 'public'. We enable our trust model for resharing by eliminating this constraint. Since Diaspora* does not allow users to specify categories for resharing and rather delivers a reshared message to all users who are related to the resharing user, we simulate the categories for a reshare by taking all categories of the resharing user. Technically, we implemented this as a patch to Diaspora* (version 0.5.3.1) consisting of 22 deleted and 20 inserted lines of code.

4.3 The Prototype

We implemented CReDiC as a policy for the CliSeAu tool. This policy specifies one EC for each service provider (called "pod" in Diaspora*). The ECs run at the

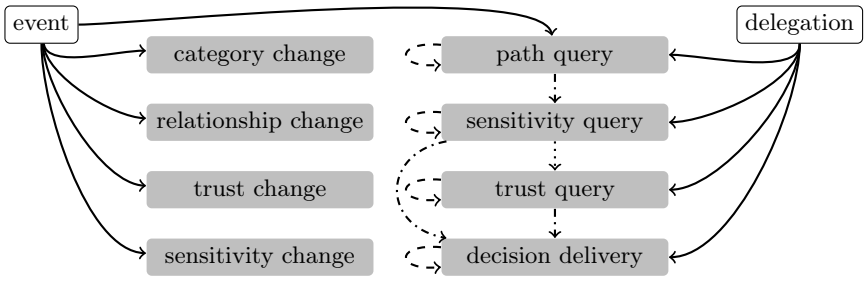

Figure 3. Modular architecture of controllers based on eight micro-policies

respective providers and are responsible for controlling the reshares performed by users at that provider. Notably, the ECs establish the same decentralized architecture as the DOSN and neither introduce any centralized component nor impose requirements on users' client software.

For controlling reshares, CReDiC specifies one method of Diaspora* to be intercepted. When a call to this method is intercepted by an EC at runtime, this EC extracts, from the arguments passed to the method, the user u who attempts to reshare a message as well as the message m to be reshared. With this information, the EC cooperates with other ECs of the DOSN as described in Sections 3.2 and 3.3 for determining whether the attempted reshare complies with the users' privacy policies.

CReDiC obtains trust values by monitoring changes of category names in profiles of users (recall that we encode the trust values in the names). It obtains sensitivity values by monitoring newly shared messages. For this monitoring, CReDiC intercepts four methods of the Diaspora* code. They allow CReDiC to keep track of dynamically changing privacy policies at the respective EC and take them into account for controlling resharing.

CReDiC is modular, consisting of eight individual components that we call "micro-policies". Four micro-policies handle changes of users' privacy policies and the storage of sensitivity values. The other four micro-policies handle the four phases for resharing. This separation yielded a low code complexity (each micro-policy is implemented in at most 41 lines of code). Figure 3 depicts the micro-policies (shaded boxes), their triggers (white boxes with solid arrows), and their temporal ordering (uncontinuous arrows). The trigger for a micro-policy is either an event of a service provider or a delegation received from another EC. The modularization allows a phase to take place at the same controller as the previous phase (dotted arrows) or at a different one (dashed arrows). The figure displays dash-dotted arrows where both cases are possible.

CReDiC globally fixes the sensitivity coefficient to $\mathsf{sc} = 0.35$. Note that the particular coordination model of CliSeAu based on delegation allows CReDiC to control simultaneously occurring reshares in an interleaved fashion, without waiting for the completion of all four phases for each individual reshare. For

securing the communication between the ECs, CReDiC utilizes CliSeAu's SSL feature. With this feature, a certificate infrastructure is automatically generated for the individual ECs such that authenticity and confidentiality of the cooperation is maintained in presence of a network attacker. Overall, the implementation of CReDiC consists of 614 source lines of Java source code and additional 184 lines of Ruby code that realizes the interface to the Diaspora* code.

4.4 Deployment and Usage

As the provider of a Diaspora* pod, one can deploy CReDiC in two steps. First, one patches the code of the pod by applying the small patch described in Section 4.2. This step can be performed automatically with the GNU `patch` tool. Second, one instruments the code of the pod with CReDiC. This step is performed automatically by invoking the extended version of CliSeAu, described in Section 4.1, with CReDiC as a parameter.

Once CReDiC is deployed to a Diaspora* pod, controlled resharing is enabled in the pod. Users specify their privacy policies via the accustomed Diaspora* web interface. Concretely, a user specifies her set of categories and the users in these categories as she would in normal Diaspora*. She can specify and update her trust in categories by modifying the trust value contained in the category name (as described in Section 4.2). In particular, users need not use further interfaces to specify their privacy policies or benefit from CReDiC's controlled resharing.

4.5 Analysis

We conducted several tests to verify the effectiveness of CReDiC. Concretely, we verified CReDiC for three policy-compliant cases of resharing: resharing a message from the pod on which it was initially shared, resharing from a different pod, and re-resharing a message involving three pods. We also verified for corresponding non-compliant cases that CReDiC successfully prohibits the resharing. That is, the tests confirmed that CReDiC effectively enforces users' privacy policies. During the tests, the ECs of all pods involved in a reshare path were online during the resharing. Since DOSN pods typically aim to be available to their users, CReDiC does not implement a fallback strategy for offline pods.

Our ReBAC mechanism and our prototype scale as follows. The number of controllers (ECs) is independent in the number of users in the DOSN and grows linearly in the number of pods. The amount of network communication performed for the reshare of a message grows linearly in the number of pods in the reshare path of the message. This is due to our first optimization described in Section 3.3. In particular, no network communication takes place between the controllers when the reshare path involves only a single pod or when a user changes her privacy policy. The computational complexity grows linearly in the length of the reshare path: for each element of the path, one lookup of a trust value and one multiplication. Because our prototype duplicates users' privacy policies in its own state and stores reshare paths, the memory required by each EC grows linearly in the number of pod users' categories and in their number of shares and reshares.

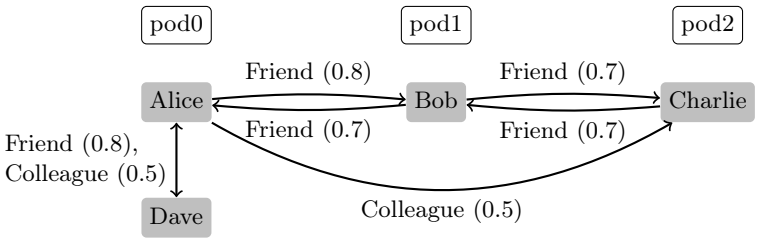

Figure 4. Example scenario for the performance evaluation

operation	*duration* CReDiC	Diaspora*	*overhead* absolute	relative
share message	231.5ms	230.3ms	1.2ms	0.52%
reshare (intra)	294.6ms	290.7ms	3.9ms	1.34%
reshare (inter)	294.3ms	281.8ms	12.5ms	4.44%
change trust	36.0ms	31.1ms	4.9ms	15.76%

Table 1. Performance evaluation results

5 Performance Evaluation

Being a mechanism that operates while the DOSN Diaspora* is running, CReDiC necessarily introduces some overhead. We evaluate how much overhead is caused with experiments in which we measure and compare the time taken by Diaspora* for resharing with and without CReDiC.

For the performance evaluation, we used three machines with Intel Quad-Core i5-4590 (3.3GHz) CPUs and 32 GB RAM. The machines ran Ubuntu 14.04.2 with 3.13.0 kernel. We ran three patched Diaspora* pods (see Section 4.2) in production mode with Ruby 2.1.1, Apache 2.4.7, and a MySQL 5.5.54 database. Four user profiles were hosted by the three pods. We measured page fetch times using `curl` 7.52.1 on an Intel Quad-Core i7-6600U (2.6GHz) with 16 GB RAM. All four machines were connected in a 1 Gbps LAN. We use a setup consisting of four users. Figure 4 displays the users (shaded boxes) and their association with the pods (white boxes). Users' privacy policies are indicated by arrows: An arrow from user u to user u' labeled with category c and trust value t represents that $rel_u(c, u')$ holds and $tv_u(c) = t$. That is, users' trust in categories is repeated on all arrows with the same source and same category label.

Table 1 shows our results. For each analyzed operation, the table contains a separate row. The first column shows the names of the operations, the second and third column show the durations of the operations in Diaspora* when CReDiC is enabled and, respectively, disabled. The fourth and fifth column show the absolute and relative overhead.

Sharing a message took 231.5ms with CReDiC enabled, compared to 230.3ms with CReDiC disabled, which corresponds to an overhead of 1.2ms (0.52%).

For resharing, we evaluated two cases: intra-provider resharing, where Alice reshares a message by Dave with her friends and colleages, and inter-provider resharing, where Bob reshares a message by Alice with his friends. For the two operations, the overhead of CReDiC ranges from 3.9ms to 12.5ms (1.34% to 4.44%). CReDiC's overhead on dynamically changing trust between users was 4.9ms (15.76%, due to the low baseline duration of 31.1ms). Each duration value in Table 1 reflects the mean of the lower 90th% of 1000 measurements [27].

Overall, CReDiC maintains a rather small performance overhead. An at first sight counterintuitive result of the evaluation is that inter-provider resharing in Diaspora* is faster than intra-provider resharing. However, Diaspora* notifies remote users asynchronously about reshared messages while users at the same pod are notified synchronously. In our scenario, inter-provider resharing notifies both recipients (Alice and Charlie) asynchronously while intra-provider resharing includes one synchronous update (Dave).

6 Related Work

Underlying our model of trust, presented in Section 2, are two main design decisions. Firstly, we model trust as scalar values ranging from 0 to 1, which can be found also elsewhere in the literature [4,17,23]. Alternatives found in the literature are models of trust based on vectors of scalars (e.g., [21]). Through vectors of scalars, individual aspects of trust such as belief and disbelief in users [21] or ability, integrity, and benevolence of users [26] can be captured in a more fine-grained fashion. We build our trust model on scalar trust values rather than vectors to give users means for quantifying their relationships while taking into account that specifying trust vectors might be a burden users refrain to take.

Secondly, our model utilizes a particular notion of trust concatenation (multiplication) and selects a single path (the reshare path) for capturing trust of an author in a resharing user. Multiplication for concatenating scalar trust values has been proposed before [4,23]. Alternative models for scalar trust values have been proposed as well. These models combine some form of multiplicative concatenation of trust with the aggregation of trust along multiple paths, for instance via weighted sums of path trust [17] or maximal path trust [23]. Further models for trust concatenation are based on trust vectors (e.g., [19]). In defining compliance with users' privacy policies based on the reshare path and no further paths between users, we see two advantages: reduced complexity and context-dependence. By context-dependence we mean that we consider the trust along the list of users who have actually seen and reshared the message, rather than users' reputations. That is, our choice of trust value reflects the notion of decision trust, which by definition is associated with a situational context. Further validation of our model or comparison to other models, e.g., by means of user studies, are beyond the scope of this article.

In our scenario, authors of messages are the sole owners of their messages. Multiparty access control (e.g., [20]) is outside the scope of this article.

The desire to control sharing and resharing has lead to the proposal of several centralized approaches. Fong et al. [6,13,14] propose a model of OSNs, a ReBAC model, and a language for expressing ReBAC policies. Relationships between users are modeled as binary relations on users. The policy language is a modal logic on the relations of the OSN that allows specifying constraints on resharing and subsequent distribution. The ReBAC mechanism for OSNs by Carminati et al. [8] enforces privacy policies of authors that can specify the maximum length of reshare paths, the minimal concatenated trust value, or relationship categories. The access control is shared between the requesting user, who provides a proof of being authorized to access the resource, and the resource provider, who checks the proof. Virtual Private Social Networks [3,9] are social networks built on centralized OSNs like Facebook but achieve privacy of user information at the client-side via a browser extension. This line of work focuses on controlled sharing of messages, not controlled resharing. SCUTA [24] is a usage control mechanism for centralized control of sharing in OSNs. The mechanism controls users' client-side operations, such as viewing, saving, and printing content.

Mechanisms for DOSNs have also been proposed. Albertini et al. [1] propose an access control mechanism for cloud-based OSNs. The proposed mechanism supports resharing but introduces centralized components, KMS and RMS, for storing keys and access rules. While the mechanism utilizes encryption for users' keys and access rules transmitted to KMS and RMS, colluding KMS and RMS could reveal the plain data. Bahri et al. [2] propose a mechanism for a-posteriori access control in a DOSN, which also relies on a centralized component (called TReMa). Our mechanism, in contrast, features a fully decentralized architecture. Safebook [10,11] and PeerSoN [7,5] are DOSNs for protecting privacy of user data. Both DOSNs include a mechanism for controlled sharing of messages. Controlled resharing is beyond their scope. GEM [29] is a distributed goal evaluation algorithm for datalog-like policies. The goal in our trust model (compliance according to Definition 3) is of a simpler but quantitative nature that cannot be specified as a goal for GEM. D-FOAF [23] is a distributed identity management system on top of trust relationships between users in multiple OSNs. D-FOAF computes the trust between two users by gathering trust values of all paths between requester and owner at one location. Our mechanism computes trust between two users based on a single path (the reshare path) and computes path trust in a distributed fashion to keep users' privacy policies decentralized.

7 Conclusion

We presented a novel enforcement mechanism that supports more fine-grained privacy policies for resharing of messages than popular OSNs like Facebook and DOSNs like Diaspora*. Our ReBAC mechanism enables controlled sharing and resharing of messages among users hosted at one service provider of a DOSN and also among users hosted at different providers. The mechanism enforces personal privacy policies of users inside a DOSN based on ReBAC. As usual for such access control mechanisms, malicious communication outside the DOSN is not

prevented. We demonstrated that the mechanism can be effectively implemented by a prototype for Diaspora* and showed that its performance overhead is small.

Our mechanism complements mechanisms for controlled sharing in OSNs by which authors know and explicitly specify the supposed recipients of messages.

Enabling authors to better control how their messages spread after resharing shall allow them to permit resharing more often, without uncontrollable dangers to their privacy. Thus, users can securely increase their outreach in DOSNs like Diaspora* and develop new personal connections with users who have received their messages via trusted others.

Acknowledgments We thank the anonymous reviewers for their comments and thank Sarah Ereth for her feedback at an early stage of this work. This work was partially funded by CASED (`www.cased.de`) and by the DFG (German research foundation) under the project FM-SecEng in the Computer Science Action Program (MA 3326/1-3).

References

1. Albertini, D.A., Carminati, B.: Relationship-based Information Sharing in Cloud-based Decentralized Social Networks. In: 4th Conference on Data and Application Security and Privacy. pp. 297–304 (2014)
2. Bahri, L., Carminati, B., Ferrari, E.: CARDS - Collaborative Audit and Report Data Sharing for A-Posteriori Access Control in DOSNs. In: IEEE Conference on Collaboration and Internet Computing. pp. 36–45. IEEE Computer Society (2015)
3. Beato, F., Conti, M., Preneel, B., Vettore, D.: VirtualFriendship: Hiding interactions on Online Social Networks. In: Conference on Communications and Network Security. pp. 328–336 (2014)
4. Beth, T., Borcherding, M., Klein, B.: Valuation of Trust in Open Networks. In: 3rd European Symposium on Research in Computer Security. pp. 3–18. LNCS 875 (1994)
5. Bodriagov, O., Kreitz, G., Buchegger, S.: Access Control in Decentralized Online Social Networks: Applying a Policy-Hiding Cryptographic Scheme and Evaluating Its Performance. In: 2014 International Conference on Pervasive Computing and Communication Workshops. pp. 622–628 (2014)
6. Bruns, G., Fong, P.W.L., Siahaan, I., Huth, M.: Relationship-Based Access Control: Its Expression and Enforcement Through Hybrid Logic. In: 2nd Conference on Data and Application Security and Privacy. pp. 117–124 (2012)
7. Buchegger, S., Schiöberg, D., Vu, L., Datta, A.: PeerSoN: P2P Social Networking: Early Experiences and Insights. In: 2nd EuroSys Workshop on Social Network Systems. pp. 46–52 (2009)
8. Carminati, B., Ferrari, E., Perego, A.: Enforcing Access Control in Web-based Social Networks. Transactions on Information and System Security 13(1) (2009)
9. Conti, M., Hasani, A., Crispo, B.: Virtual Private Social Networks and a Facebook Implementation. Transactions on the Web 7(3), 14:1–14:31 (2013)
10. Cutillo, L.A., Molva, R., Strufe, T.: Safebook: A Privacy-preserving Online Social Network Leveraging on Real-life Trust. Communications Magazine 47(12), 94–101 (2009)

11. Cutillo, L.A., Molva, R., Strufe, T.: Safebook: Feasibility of Transitive Cooperation for Privacy on a Decentralized Social Network. In: 10th International Symposium on a World of Wireless, Mobile and Multimedia Networks. pp. 1–6 (2009)
12. Datta, A., Buchegger, S., Vu, L., Strufe, T., Rzadca, K.: Decentralized Online Social Networks. In: Handbook of Social Network Technologies and Applications, pp. 349–378 (2010)
13. Fong, P.W.L.: Relationship-Based Access Control: Protection Model and Policy Language. In: 1st Conference on Data and Application Security and Privacy. pp. 191–202 (2011)
14. Fong, P.W.L., Anwar, M.M., Zhao, Z.: A Privacy Preservation Model for Facebook-Style Social Network Systems. In: 14th European Symposium on Research in Computer Security. pp. 303–320. LNCS 5789 (2009)
15. Gates, C.E.: Access Control Requirements for Web 2.0 Security and Privacy. In: Workshop on Web 2.0 Security & Privacy (2007)
16. Gay, R., Hu, J., Mantel, H.: CliSeAu: Securing Distributed Java Programs by Cooperative Dynamic Enforcement. In: 10th International Conference on Information Systems Security. pp. 378–398. LNCS 8880 (2014)
17. Golbeck, J.A.: Computing and Applying Trust in Web-based Social Networks. Ph.D. thesis, University of Maryland (2005)
18. Grippi, D., Salzberg, M., Sofaer, R., Zhitomirskiy, I.: The Diaspora* Project. `http://diasporafoundation.org/` (February 2016)
19. Hang, C., Wang, Y., Singh, M.P.: Operators for propagating trust and their evaluation in social networks. In: 8th International Joint Conference on Autonomous Agents and Multiagent Systems. vol. 2, pp. 1025–1032 (2009)
20. Hu, H., Ahn, G., Jorgensen, J.: Multiparty Access Control for Online Social Networks: Model and Mechanisms. IEEE Transactions on Knowledge and Data Engineering 25(7), 1614–1627 (2013)
21. Jøsang, A.: A Subjective Metric of Authentication. In: 5th European Symposium on Research in Computer Security. pp. 329–344. LNCS 1485 (1998)
22. Jøsang, A., Ismail, R., Boyd, C.: A Survey of Trust and Reputation Systems for Online Service Provision. Decision Support Systems 43(2), 618–644 (2007)
23. Kruk, S.R., Grzonkowski, S., Gzella, A., Woroniecki, T., Choi, H.: D-FOAF: Distributed Identity Management with Access Rights Delegation. In: 1st Asian Semantic Web Conference. pp. 140–154 (2006)
24. Kumari, P., Pretschner, A., Peschla, J., Kuhn, J.M.: Distributed Data Usage Control for Web Applications: a Social Network Implementation. In: 1st Conference on Data and Application Security and Privacy. pp. 85–96 (2011)
25. Mao, H., Shuai, X., Kapadia, A.: Loose Tweets: An Analysis of Privacy Leaks on Twitter. In: 10th Annual ACM workshop on Privacy in the Electronic Society. pp. 1–12 (2011)
26. Mayer, R.C., Davis, J.H., Schoorman, F.D.: An Integrative Model of Organizational Trust. Academy of Management Review 20(3), 709–734 (1995)
27. Oaks, S.: Java Performance - The Definitive Guide: Getting the Most Out of Your Code. O'Reilly (2014)
28. Paul, T., Famulari, A., Strufe, T.: A Survey on Decentralized Online Social Networks. Computer Networks 75, 437–452 (2014)
29. Trivellato, D., Zannone, N., Etalle, S.: GEM: A Distributed Goal Evaluation Algorithm for Trust Management. Theory and Practice of Logic Programming 14(3), 293–337 (2014)
30. Wampler, D.: Aquarium: AOP in Ruby. In: Aspect Oriented Software Development (2008)