A new Sharing Paradigm for the Personal Cloud

Paul Tran-Van[1,2,3], Nicolas Anciaux[2,3] and Philippe Pucheral[2,3]

[1] Inria Saclay-Île-de-France, 1 rue d'Estienne d'Orves , 91120 Palaiseau, France
[2] DAVID Lab., University of Versailles, 45 av. Etats-Unis, 78035 Versailles, France
[3] Cozy Cloud, 158 rue de Verdun, 92800 Puteaux, France
first_name.last_name@inria.fr

Abstract. Pushed by recent legislation and smart disclosure initiatives, personal cloud solutions emerge and hold the promise of giving the control back to the individual on her data. However, this shift leaves the privacy and security issues in user's hands, a role that few people can properly endorse. Considering the inadequacy of existing sharing models, we advocate the definition of a new sharing paradigm dedicated to the personal cloud context. This sharing paradigm, called SWYSWYK (Share What You See with Who You Know), allows to derive intuitive sharing rules from the personal cloud content, to self-administer the subjects and the sensitive permissions, and to visualize the net effects of the sharing policy on the user's personal cloud. We then propose a reference architecture providing the users tangible guarantees about the enforcement of the SWYSWYK policies. An instance of this architecture has been implemented on top of an existing personal cloud platform to demonstrate the practicality of the approach.

Keywords: Personal data sharing, personal cloud, self-administered policies.

1 Introduction

Today, *smart disclosure* initiatives are pushed by legislators (e.g., EU General Data Protection regulation [1]) and industry-led consortiums (e.g., Blue Button for medical records in the US, Midata in the UK, MesInfos in France) in order to enable individuals to get back their personal data from companies or administrations that collected them. The Personal Cloud paradigm emerges (e.g., Cozy Cloud, ownCloud, SeaFile, Databox) and holds the promise of a Privacy-by-Design storage and computing platform where each individual can gather her complete digital environment in one place and share it with applications and users under her control.

However, this gravity shift of data management from organizations to individuals raises new fundamental issues. One of the founding principles of the Personal Cloud paradigm is to enable individuals making sovereign decisions about the sharing of their data, i.e., administering sharing rules to regulate data dissemination. This task is difficult for regular users who are not database administrators nor security experts. Indeed, the main existing access control models are not adapted to the personal cloud context. Existing solutions are either geared towards central authorities, allowing

them to properly define users, roles and privileges thanks to robust models (e.g., RBAC, MAC, ABAC or TBAC [2]) or suggest decentralized tools to let individuals manually define their sharing preferences (e.g., thanks to PGP, Web of Trust models or Friend of a Friend (FOAF) dissemination rules [3]). The former approach is adapted to a centralized database and requires a deep expertise in terms of administration and security. The latter puts all the cognitive load of defining sharing rules to the user while providing tools of limited expressive power (privileges are declared manually on a user-resource case-by-case basis). Some solutions like [4] and [5] try to exploit machine learning techniques to automatically infer the best sharing policies but they can lead to unexpected data leakage when the classification goes wrong.

We thus advocate the definition of a new sharing paradigm dedicated to the personal cloud context. How could regular users share the recent information obtained from their quantified-self appliances with medical practitioners of their acquaintance? How photos of an excursion can be shared with the relatives appearing in these same photos? How to avoid personal pictures to be shared with working colleagues?

Our proposal builds upon the transversal nature of the content of a personal cloud and makes easy and intuitive the definition and administration of sharing policies. The personal cloud content intrinsically describes the individual's acquaintances under different forms (e.g., contact files, agendas, identity pictures, address book entries, etc.). Conversely, acquaintances are associated with pieces of information in the user's space (e.g., photos on which a friend appears). New sharing models should be able to map personal data to acquaintances (or subjects) and exploit their links with the stored documents (or objects) to produce authorizations satisfying users' sharing desires such as those expressed above. Interesting and common sharing rules could also be published and adopted by the members of a community of interest. Beyond the definition of the sharing policy, the sharing paradigm must provide means to the personal cloud owner to easily understand the net effects of a sharing policy, identify suspicious permissions and sanitize the sharing policy accordingly, and finally, to trust the way the policy is practically enforced.

In this paper, we make the following contributions:

- we propose a new sharing paradigm called SWYSWYK (*Share What You See with Who You Know*) which allows to automatically derive intuitive sharing rules from a personal cloud content, to let each user visualize the net effects of these rules on her own personal cloud and to finely customize data sharing according to their own privacy concerns,
- we introduce a reference architecture for the SWYSWYK paradigm which helps the user to administer her sharing policy and provides tangible guarantees about its enforcement, and show the feasibility of our approach with an implementation combining an existing personal cloud platform and a secure personal device.

The rest of the paper is organized as follows. Section 2 presents related works and Section 3 derives from them a problem statement. Section 4, 5 and 6 are devoted to the aforementioned contributions, and Section 7 concludes.

2 Related Works

This section positions existing access control approaches with respect to the personal cloud context, in order to introduce our problem statement in the next section.

Traditional database access control models. The access control management in databases is well established and models like DAC, RBAC and MAC [2] are widely supported. The question is whether or not such models can apply to the personal cloud context. These models actually share the following characteristics: (1) the access control administration is a complex and critical task usually handled by a security expert; (2) the applicative logic, the users and their roles are identified at an early stage of the information system design, so that the access control policy is part of the database schema definition and (3) these models are integrated in standards like SQL and benefit from the expressive power of such languages. However, in a personal cloud context, nothing of the above still holds. First, the administrator of a personal cloud is the owner herself and, apart from rare exceptions, is not a technical expert. Second, usages are difficult to predict because new appealing apps are produced at a high rate and so are the interactions between users. Third, no central authority delivers a common framework to identify unambiguously subjects and objects in a potential access control rule and no universal standard exist to define and manipulate them. These reasons make traditional access control models not suitable for the personal cloud context.

Access Control in a Decentralized Setting. The decentralization aspect raises new challenges starting with users' identification and authentication. To avoid any confusion between users, let us call *owner*, the owner of a personal cloud and *subjects*, the users willing to interact with the personal cloud owner. Web of Trust (WoT) models have been investigated, allowing subjects to authenticate thanks to their public key and are identified through social attestations [6] or public profiles [7], to which access rights are associated and defined by the owner. It requires the owner to manually assign the authorizations, for each subject and for each object, which can quickly become tedious and error-prone. [8] relies on a centralized trusted party for the authentication. A second challenge is enforcing the access control rules, meaning preventing confidentiality attacks over data to be shared. [9, 10, 11] focus on data encryption in untrusted clouds, while [12] presents a decentralized alternative to social networks based on ABE encryption. [13, 14] use obfuscation schemes, where data is scrambled for the former and substituted for the latter. Access control policies are implemented here by means of encryption but, again, this requires the owner to manually define who can access which data on a case-by-case basis. Thus, the cognitive load on the owner is such that it often leads to consider access control as an intractable burden, letting desperate owners define far too permissive policies [15].

User-friendly AC Administration. Several contributions have been proposed to ease the access control administration by allowing the declaration of logic-based sharing rules. [16, 17] propose rule frameworks based on fixed attributes, but do not cope well with the versatile personal cloud context. Negative policies are also supported to deal with exceptions that can occur in sharing policies. [18] defines an SQL-based language to let applications create queryable views on the shared data, and transmit capabilities to granted subjects. Some works propose to use existing user’s social

relationships as a convenient way to facilitate policies declaration. In [19], subjects are granted access depending on their FOAF relationship properties, such as type, graph depth, and a computed trust value. [20] adds web scraping on social networks to retrieve common social events attendance. Finally, [4] and [5] focus on machine learning techniques to automatically infer the best sharing policies. [4] takes small manual input from the user and exploits its data on social networks, including profile characteristics and relationships, to extract sharing communities patterns. [5] uses an image classification module based on content and metadata from large images set, and analyze the owner privacy preferences to predict the sharing policy for each of her uploaded photos. These works both claim a good accuracy in the predicted policies and can greatly ease the access control administration, but they can also lead to unexpected data leakage when the classification goes wrong. It is also not certain that owners would accept to let an opaque algorithm scrutinize their data and their social relationships.

3 Problem Formulation

This state of the art highlights two major difficulties that need to be circumvented when designing a sharing model for the personal cloud context : (1) *The owner is the weakest link of the security chain*: she is de facto the administrator of her personal cloud platform, but it is illusory to expect her gaining expertise and spending time to administer subjects, secure her personal cloud against all forms of attacks or use tricky protocols to exchange cryptographic secrets with partners; and (2) *personal cloud usage is versatile and volatile*: while traditional information systems are built to support well identified services invoked by well identified users and applications, the personal cloud world favors opportunistic usages and unpredictable interactions between users. We derive from these statements three major requirements for a sharing model dedicated to the personal cloud:

- *Enlighten empowerment*. User's empowerment should be enlightened, meaning that the effects of all owner's decisions must be perceivable and understandable by herself. We advocate the integration of a SWYSWYK (pronounce Swiss-week) principle (Share What You See with Who You Know) in the definition of access control policies. Roughly speaking, this means that the model should provide means to derive intuitive sharing rules from the content of personal cloud documents and let the owner visualize the net effects of these rules on her personal cloud.
- *Self-sustaining administration*. In the line of the SWYSWYK principle, the administration of subjects must be intuitive and (quasi) automatic, derived from the content of personal cloud documents and from regular actions performed by the owner on these documents. As well, the owner should be offered simple means to finely administer the effects of the sharing rules according to her own privacy concerns.
- *Tangible enforcement*. To give substance to the SWYSWYK principle, the logic of the reference monitor enforcing the sharing policy must itself be perceivable and understandable by the owner and the platform implementing this logic must be

trusted by her. Typically, we don't believe that a regular user can figure out the results of neither a powerful solver taking as input a set of positive and negative sharing rules nor that she can trust a remote machine or her vulnerable computer to run this logic. A side effect of the expected extreme simplicity of our model is that the reference monitor could be embedded in a tamper-resistant personal device kept in user's hand, thereby providing a physical element of trust.

Of course, these requirements are not enough to solve all the problems identified above, but we believe they can lead to a more affordable and trusted sharing process.

4 SWYSWYK Model

As many existing proposals, SWYSWYK is a rule-based model, but our goal is not to propose yet another more expressive rule-based access control model. The originality of SWYSWYK lies in a few elementary core principles, which can be summarized as follows: *documents are rules* and *subjects and objects are documents*. The combination of these principles gives birth to an access control model tackling important requirements of the personal cloud context. We introduce here the model baselines and semantics. Operational aspects are discussed in the next section.

4.1 Model baseline

Let us first illustrate the impact of *enlightened empowerment* in the access control declaration. Sharing a picture with people appearing on it, sharing a document minutes with the meeting attendees or an agenda entry with the corresponding relatives should be straightforward to express. Subsequent permissions can be "derived" from the documents' content, leading to our first core principle: *documents are rules*. The subjects directly concerned with a document, also called *identifiees* [21], should also be extracted from the document content and be identifiable as such to enter in the rule definition. We call "reflexive sharing rules" the rules based on this principle. A corollary of this is that each subject should correspond to a viewable personal cloud document (i.e., *subjects are documents*), e.g., contact record, resume or picture. As well, whether the result of a complex treatment over a set of documents needs to be shared (e.g., an unintelligible computation over a set of smart meter measurements), this treatment must output a viewable shared document (e.g., a curve of consumption). The combination of *documents are rules* and *subjects and objects are documents* gives substance to the '*Share What You See with Who You Know*' paradigm. The impact of *self-sustaining administration* is also paramount, automating the subject declaration and maintenance (e.g., sharing meeting minutes with attendees could automate the creation/updates of subjects). Regarding the *tangible enforcement* requirement, the decision process must remain as simple as possible. And in any case the administration of the sharing rules must remain in line with the privacy concerns of the owner.

We show next how these principles can be integrated in a simple sharing model, and let open the discussion of their integration in a more powerful/expressive model or in traditional existing ones. Hence, we do a set of simplifying assumptions. First,

our model relies on a closed policy (every action not explicitly granted is denied). Actions are CRUD operations on documents in the personal cloud. The model supports only authorizations (positive rules) and allows the owner to post-filter the Access Control List (ACL) produced when exceptions need to be declared. Consequently, there is a direct translation between sharing rules and sets of ACLs. An action *a* is granted to subject *s* on document *d* iff $(s,d,a) \in ACL$ and is denied otherwise. The model is by construction *consistent* (the decision is unique), *complete* (the decision always exists) and can finally be evaluated in *logarithmic time*.

4.2 Sharing Model Semantics

Reflexive sharing rules implement the *documents are rules* principle, and are thus considered as first-class citizen rules in SWYSWYK. Such rules express the sharing of documents with subjects directly concerned with those documents. We introduce below a set of notations required to define the semantics of reflexive sharing rules.

D: set of all documents in a personal cloud (by extension, set of all *DocId*).
S: set of all subjects in a personal cloud (by extension, set of all *SubjectId*).
A: set of actions which can be performed on elements of *D*.
Q: set of qualifications which can be expressed on elements of *D* with the host language of the personal cloud (we do not make any assumption on this language).
IT: set of identification traits uniquely linked to any element of *S* (e.g., ssn, <lastname, {firstname}>, pseudo, phone n°, docId of a contact entry...).

Let us now consider the following relations and functions:

$ACL \subseteq S \times D \times A$: set of Access Control Lists.
Allowed: $S \times D \times A \rightarrow \{true, false\}$: characterizes the access control
Allowed (s, d, a) = *true* iff $(< s, d, a > \in ACL)$
= *false* otherwise
SI: $S \rightarrow IT$: delivers all known identification traits of any element of *S*.
MatchS: $\mathcal{P}(IT) \times \mathcal{P}(IT) \rightarrow \{true, false\}$ where $\mathcal{P}$ is the powerset of a set. *MatchS* evaluates the pairwise correspondence between subjects by comparing *ITs*.
MatchS(*ident1*, *ident2*) = *true* iff $(ident1 \cap ident2 \neq \varnothing)$
= *false* otherwise
IsS: $D \rightarrow S \cup \Phi$: gives the unique element of *S* characterized by a document or Φ (absence of value) if the document does not correspond to any subject. *IsS* is surjective, i.e., $(\forall s \in S, \exists d \in D \,/\, IsS(d)=s)$, meaning that each subject is represented by an existing (viewable) document in the personal cloud.

Two additional functions are independent of the access control logic and could be provided by communities or personal cloud providers. For the sake of genericity, they are considered as user-defined functions (UDFs) in the model:

DI: $D \rightarrow \mathcal{P}(IT) \cup \Phi$: delivers the identification traits (potentially from multiple subjects) contained in a document or Φ. Example of a *DI* could be a facial recognition function for pictures or a function extracting identification traits (e.g. email addresses, phone numbers) from a text document.
Filter: $D \times Q \rightarrow \{true, false\}$: evaluates whether $d \in D$ matches a qualification *Q*. *Filters* are used to form subsets of documents satisfying a given criteria. Filters

are personal cloud platform dependent. They are assumed to select documents either based on their content or on metadata (e.g., type, creator, date, tags) attached to each document by the personal cloud.

Reflexive sharing rules. Thanks to the above notations, reflexive sharing rules can be expressed as follow:

$ACL \leftarrow \{(s,d,a) \in S \times D \times A \,/\, Filter_1(d,Q) \wedge MatchS(DI(d), SI(s)) \}$

Examples of reflexive sharing rules follow for illustration purpose:

Example 1. Share the minutes of meetings with the corresponding attendees:

Q: docName like '../Meetings/minutes-%.doc'

DI: extract attendee names from a minute document

Example 2. Share the photo gallery 'MyBirthday 2016' with people who appear in the pictures:

Q: docType = '.jpg' ∧ tagGallery = 'MyBirthday 2016'

DI: face recognition function from .jpg files

For this example, profile pictures linked to contact forms could be used to recognize the faces with a trained model.

It is likely that reflexive rules apply additional restrictions over the subjects identified in documents, leading to the following more complete definition of reflexive rule:

$$ACL \leftarrow \{(s,d,a) \in S \times D \times A \,/\, Filter_1(d,Q) \wedge MatchS(DI(d), SI(s)) \wedge \exists d' \in D,\ Filter_2(d',Q') \wedge (IsS(d')=s) \}$$

Example 3. Share the minutes of a meeting held in Paris on March 12th 2017 with the members of my team attending the meeting:

Q: docId = /lib1/lib2/minutes-Paris-120317.doc

Q': docType = '.vcf' ∧ tagStatus = 'Team member'

DI: extract attendee names from a minute document

Basic sharing rules. Non reflexive rules can also easily be supported by the model. In particular, the usual rules sharing a selection of documents ($Filter_1$) with a selection of subjects ($Filter_2$), called basic rules, are expressed as follows:

$$ACL \leftarrow \{(s,d,a) \in S \times D \times A \,/\, \exists d' \in D,\ Filter_2(d',Q') \wedge IsS(d')=s \wedge Filter_1(d,Q)\}$$

Example 4. Share document PCloud/MedicalFolder/Myheartbeat with doctors:

Q: docId = PCloud/MedicalFolder/Myheartbeat

Q': docType = 'contact' ∧ tagStatus = 'Physician'

Example 5. Share the photo gallery 'MyNewPaintings' with my friends:

Q: docType = 'photo' ∧ tagGallery = 'MyNewPaintings'

Q': docType = 'contact' ∧ tagStatus = 'Friend'

Example 6. Share my calorie intakes of the last 3 months with my family doctor:

Q: docType = '.xls' ∧ tagContent = 'calorie' ∧ Date ≥ 'CurrentDate – 3 month'

Q': docType = '.vcf' ∧ tagStatus = 'Family doctor'

4.3 Administration Model

The subject declaration and maintenance must be (quasi) automatic and the owner should be able to apply her own privacy concerns while administrating rules and per-

missions. We first introduce the notion of *rule consistency*, which concretizes the fact that the effects of all rules can be visualized. Then, we show how owner's specific privacy concerns are supported by means of *exceptions* without introducing any complexity. Third, we explain how *subject administration* can be performed transparently as a constituent part of the sharing rules.

Rules Consistency. A SWYSWYK sharing rule is said well-formed iff it produces only ACLs involving viewable documents shared with recognizable subjects. More notations are introduced here related to the SWYSWYK-like administration.

- $DV \subseteq D$: the subset of documents being viewable (in the interpretable sense) by the owner. In other words, $DV = \{d \in D \,/\, \exists Viewer(d)\}$ with *Viewer* an application trusted by the owner (e.g., certified by an authority or a community of users) which delivers an interpretable view of the document to the owner (e.g., potentially transforms a binary format into a text, an image or a graphic).
- $DS \subseteq DV$: set of viewable documents characterizing a unique subject (e.g., a vCard file, a resume, a photo labeled with a subject ID).
- $DI \subseteq D$: set of documents containing identification traits of subjects or identifiees (e.g., an agenda, a photo of a group of people, the meeting minutes).
- SR: set of all sharing rules defined by the owner.

Based on these notations, the SWYSWYK sharing rules are well-formed iff $\forall sr \in SR, \forall acl \in ACL, acl.d \in DV \wedge acl.s \in DS$. Any *acl* which does not satisfy this condition will be filtered out.

Rules Exceptions. We enacted as a design principle the fact to consider a closed policy and to allow only the declaration of positive rules (i.e., authorizations). We exclude negative rules (i.e., interdictions) because we consider that a common owner cannot easily figure out the output of a policy mixing potentially conflicting positive and negative rules, making the cognitive cost of administering his policy out of reach. Thus, instead of enriching the model semantics to capture authorizations and interdictions, we simply give the owner the ability to filter out a posteriori some permissions which may hurt her privacy (considered as exceptions). Hence, the resulting logic of the reference monitor remains straightforward, thereby complying with the *tangible enforcement* property. As explained below, an administration GUI is devoted to this task.

An ACL can be considered suspicious either because it involves a sensitive subject (e.g., my manager) or a sensitive object (e.g., a compromising picture or a part of my medical folder) or because the association between a particular subject and object may itself be compromising (e.g., I'm not ready to share all my holiday pictures with my colleagues, even if I trust them and if most of these pictures are not sensitive). Based on such information, we consider three types of enquiry queries, respectively targeting sensitive subjects, sensitive objects and sensitive associations between them, where $ACL^?$ denotes a set of newly generated ACLs:

- $What(Q_S, A) \rightarrow \{(s,\{(d,a)\}) \,/\, (s,d,a) \in ACL^? \wedge s \in Q_S(S) \wedge a=A\}$: identifies, for each selected (sensitive) subject, the new set of action a they are granted to perform on which documents d (e.g., "which new documents can be seen by my boss?").

Who(Q_D, A) → {(d,{(s,a)}) / (s,d,a)∈$ACL^?$ ∧ d∈Q_D(D) ∧ a=A}: identifies, for each sensitive document d, the new set of subjects s with granted action a on them (e.g., "which new subjects have a read access to my medical records?").

Which(Q_S, Q_D, A) → {(s,d,a) / (s,d,a)∈$ACL^?$ ∧ s∈Q_S(S) ∧ d∈Q_D(D) ∧ a=A}: identifies new ACLs combining a selection of (sensitive) subjects and documents (e.g., "which new authorizations my colleagues have on my family photos?").

An administration GUI lets the data owner declare *suspicion clauses*, that is select sensitive subjects (i.e., Q_S(S) clauses), sensitive documents (i.e., Q_D(D) clauses) and compromising association (i.e., pairs of (Q_S(S), Q_D(D)) clauses). The inquiry queries *What*, *Who* and *Which* based on these clauses are *watchdog triggers* evaluated on any new set of ACLs produced by sharing rules. Certain suspicious ACLs are then identified and need the owner's validation that can be easily done through the GUI. This guarantees that no unwanted disclosure will be done on sensitive data.

Subjects Administration. The administration of subjects is usually one of the most cumbersome task for a owner. The objective here is to make this task as transparent as possible, extracting the subject definition from the documents themselves and from the rules declaration. Let us first detail the structure of *S*. *S* can be represented by a table of schema (*Sid*: *S*, *Did*: $\mathcal{P}$(*DS*), *It*: $\mathcal{P}$(*IT*), [*Ct*: *CONTACT*], [*Auth*: {(*AUTH, credential*)}]), where *Sid* is the subject identifier, *Did* is a (set of) document identifier(s) referencing document(s) representing this unique subject on personal cloud, *It* is the (set of) identification trait(s) of this subject, *Ct* and *Auth* are optional –personal cloud dependent– information required to notify and authenticate the granted subjects. New subjects are registered in *S* as side-effects of the function *IsS*(*d*) (algorithm below).

Function *IsS*

Input: $d \in D$

Output: $s \in S$ if d corresponds to a recognized subject or Φ otherwise

1. **if** $\exists\, s \in S$ / *MatchS*(*DI*(*d*), *SI*(s)) **then**
2. $s.It \leftarrow s.It \cup DI(d)$
3. **else if** *RegistrationAgreement*( ) **then**
4. $s \leftarrow$ *NewSid*( )
5. $S \leftarrow < s, d, DI(d), [Ct(s)], [Auth(s)] >$
6. $DS \leftarrow d$
7. ***else*** $s \leftarrow \Phi$
8. ***return*** s

Algorithm 1. Function IsS

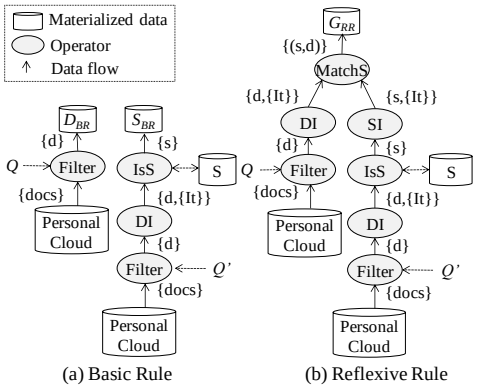

Fig. 1. ACL Production.

IsS is automatically invoked (1) each time *DS* documents are created or updated in the personal cloud (e.g., contacts documents, resumes, etc., as determined by the personal cloud platform) and (2) each time a new rule invoking *IsS* is defined. If at least one identification trait present in document *d* matches an existing subject *s*, *s* is returned and *s.It* is potentially enriched with the other identification traits present in *d*. If no correspondence is found, the owner may be asked on the fly to accept the registration of a new subject based on *d* content. Otherwise, *IsS* fails and Φ is returned.

Hence the set of subjects automatically grows along document insertions and rule declarations with minimal owner interaction. The owner may however wish to disambiguate from time to time the content of *S* through an administration GUI (e.g., merge duplicates in situations like *'John Doe'∈s.It* and *'john@doe.io'∈s'.It*). Smart mechanisms may help, like relying on FOAF [3] and trying to parse the public pages of friend users to discover their identification traits, but this is let for future works.

5 Operational Semantics

We discuss here operational and architectural baselines in SWYSWYK, which are critical to enable a secure implementation of the model.

5.1 SWYSWYK Operational Baselines

The creation, maintenance and evaluation of a set of SWYSWYK permissions is as follows: (1) the owner creates sharing rules and *suspicion clauses* to be applied on her personal cloud; (2) a *rule translator* translates the selected rules into candidate and suspicious ACLs; (3) the owner checks the produced ACLs at will and accepts or rejects the suspicious ones using the *administration GUI*; (4) the *reference monitor* authenticates subjects and evaluates *Allowed*(*s*,*d*,*a*) calls to *true* or *false* and delivers the requested documents accordingly; and (5) new subjects are automatically integrated in the owner's personal cloud at document insertion and rule translation time.

Steps (3) and (5) are unusual in the access control management. Step (5) contributes to the *self-sustaining administration* property and step (3) gives form to the *enlighten empowerment* property by pushing the owner to check the net effect of sharing rules. The administration GUI helps the owner turning each new $ACL^{?}$ element either into an *accepted* permission (called ACL^{+}) or an *exception* (called ACL^{-}). The ACL^{+} set is the sole to be considered by the reference monitor when making access decisions. ACL^{-} is similar to exceptions of a positive rule and is materialized to automatically filter out unexpected authorizations in subsequent runs of the rule translator, without owner intervention. Hence, in contrast to rule-based reference monitors, the logic of a SWYSWYK reference monitor can be trivially understood by anyone (operation *a* on *d* is granted to *s* iff (*s*,*d*,*a*)∈ ACL+) and contributes itself to *enlighten empowerment*.

5.2 Sharing Rules Constructs and Management

Our model is based on the materialization of all ACLs, which makes the evaluation of the *Allowed* function trivial and enables the user to visualize and filter the reference monitor effects. As shown on Fig. 1, five physical operators are required to express any basic or reflexive sharing rule, namely *Filter*, *DI*, *SI*, *IsS*, *MatchS*. The semantics of these operators are equivalent to their functional counterparts presented in Section 3 and thus are not recalled. The main difference is that each operator implements the corresponding function in a set-oriented way. For instance, *Filter* operator applies to all documents of *D* and returns the subset of documents satisfying condition *Q*. The

flow of data consumed and produced by the operators is presented in Fig. 1 for the translation of basic and reflexive rules into ACLs.

At declaration time, the rule translator must evaluate a new sharing rule over all documents of the personal cloud. For a basic rule, it applies a filter at the leaf of each branch (i.e., selection of targeted subjects on the right branch and of targeted documents on the left branch). Then *DI* operator extracts the list of *ITs* from the targeted subject documents while *IsS* tries to match these *ITs* with the subjects already registered in *S*. As discussed in Section 4.3, *IsS* operator may have side-effects to dynamically populate *S* when unknown subjects are encountered. Finally, for a basic rule *BR*, the right branch feeds the S_{BR} structure while the left branch feeds the D_{BR} structure registering the produced (candidate) ACLs. The operator tree of reflexive rules follows the same logic, except that left and right branches must be joined on subject *ITs* and produce (candidate) ACLs on the G_{RR} structure.

Each time a new document *d* is inserted into the personal cloud, the filters of each branch of all rules are evaluated against *d* to check whether new candidate ACLs can be produced. While this cost is rather low for basic rules, the *MatchS* operator at the root of the reflexive rule tree may incur a full re-scan of the left branch on personal cloud (e.g., a subject *s* inserted at time t_2 may match with a document *d* inserted at time $t_1 < t_2$, while this association was not detectable at time t_1). We introduce an additional structure registering pending reflexive associations between subjects and documents to alleviate this cost (see Section 6). Whether a document *d* is deleted from the personal cloud, any entry referencing *d* in S_{BR}, D_{BR} and G_{RR} must be removed.

5.3 Reference Architecture and Enforcement

Although formally proving the security of an architecture for SWYSWYK is out of the scope of this paper, this section introduces an abstract architecture implementing the model while providing the *tangible enforcement* property. We consider an architecture made of (1) an *untrusted environment* (*UE*) on which no security assumption is made, (2) an *isolated environment* (*IE*) on which code can run with the guarantee that its execution is isolated from UE and (3) a *Secure Execution Environment* (*SEE*) which protects data and code against snooping and tampering.

Since no specific security assumption can be made on the personal cloud platform, it is part of the UE. All the documents of the personal cloud are thus stored encrypted and can be decrypted only by the reference monitor, which acts as an incorruptible doorkeeper for the personal cloud. Whenever an *Allowed(s,d,a)* request is evaluated to *true*, document *d* is decrypted in the SEE before being delivered to subject *s*. The reference monitor resides in the SEE and given its simplicity, can be hosted in many kinds of tamper-resistant SEE (e.g., SIM cards in smartphones, secure personal tokens [22, 23]). Data structures like *SR*, the set of sharing rules activated by the owner, *S*, the set of subjects in relation with the owner, and $ACL^?$, ACL^+ and ACL^- are stored in the SEE.

The rule translator updates these internal data structures. *IsS*, *MatchS* and *SI* are internal operators running inside the SEE. *Filter* and *DI* must be extensible (e.g., integrate existing libraries) and thus cannot be stored in the SEE. However, they need to

access large portions of the personal cloud, leading to potential risks of information disclosure if observed or corrupted. Thus, *Filter* and *DI* are running in *isolated containers*[1] in the IE. The *Administration GUI* and the document *viewers*, which involve interactions with the owner, also run in IE.

Several physical instances of this architecture are possible. The experimental platform presented in the next section is an extreme case, but other target architectures could be envisioned. For example, a certified hypervisor running on top of an Intel SGX processor [24] on the personal computer itself can provide a logical implementation of IE. Smart devices equipped with a SIM card (embedding the reference monitor), a flash memory card (embedding the encrypted internal data structures) and an ARM Trustzone processor [25] (running UDFs and administration tools) are other interesting options.

6 Experimental Platform

We consider here a specific instance of the SWYSWYK architecture combining Cozy (open-source personal cloud, see: https://cozy.io/) and PlugDB (developed at Inria, see: https://project.inria.fr/plugdb/). The personal device has a 3GHz Intel Xeon E5-1660 CPU, 8 GB of RAM and a 500 GB 10.000 RPM hard drive. PlugDB uses a STM32F417GH6 MCU with a 168 MHz ARM Cortex M4 CPU, 192 KB of RAM and 1 MB of NOR storage. The Raspberry Pi 3 has a 1.2GHz ARMv8 CPU and 1 GB of RAM. We use an external UHS-I microSD card of 16 GB for both SEE and IE. A video of the experimental platform and its usability is available[2].

The Cozy system runs on a personal device linked to the network (Internet) and represents the *UE*.

The secure parts of the reference monitor runs on PlugDB (the *SEE*), communicates with *UE* in WiFi (IEEE802.11n) and with *IE* in USB2.

The admin. GUI and UDFs are installed on a Raspberry Pi without any network connection, which represents the *IE*.

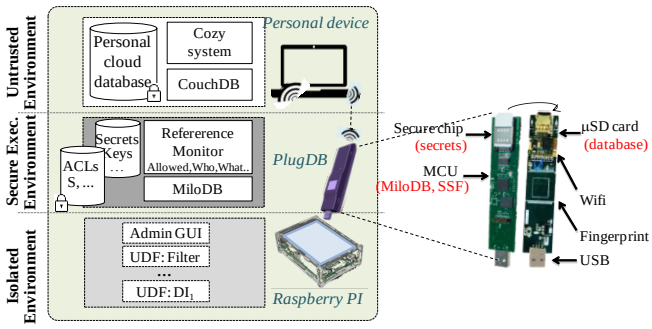

Fig. 2. Experiment platform: software and hardware (right).

Environmental cost. The environmental cost to insert a new document in the personal cloud is pictured in Fig.3(left). The 4 components of this cost are: (1) transfer cost of the document between UE and SEE (in cleartext from UE to SEE and in encrypted form back), (2) document encryption cost in SEE, (3) insertion of the en-

[1] In practice, isolated containers can be implemented using a dedicated hardware platform (physical isolation), an hypervisor or a microkernel. Recent hardware advances propose an hardware support for running isolated code, e.g., using ARM Trustzone [2] or SGX processors [9].

[2] http://wanda.inria.fr/CIKM/cikm.ogg

crypted document in Cozy and (4) transfer cost of the cleartext document from SEE to IE (which applies *Filter* and *DI* on sharing rules). Step 4 (USB transfer) can be performed in parallel with the other steps, explaining why we isolate that cost in Fig. 3.

Rules and data sets. In Table 1, we introduce the rules and data sets used in the experiments to measure the performance at rule translation time (i.e., initialization cost incurred by a rule creation and maintenance cost of a rule when new documents are inserted). We define four different rules representative of *basic* and *reflexive* rules, with *Big* and *Small* output sizes in terms of the number of produced ACLs.

Table 1. Sharing rules considered in the experiments. E.g., Big BR is a big basic rule qualifying 1.000 documents on the predicate *type='cardio'* and 10 subjects on *type='health community'* resulting in 10K ACLs (this rule corresponds to a quantified-self context).

Rule name	***Filter on D***	***Filter on S***	***#results: D, S, ACLs***
Small BR	type = 'directory' & name = 'team'	type = 'contact' & group = 'team'	10, 5, 50
Big BR	type = 'cardio'	type = 'health community'	1.000, 10, 10.000
Small RR	type = 'note'	type = 'contact' & group = 'lab'	10, 30, 50
Big RR	type = 'album' & tag = 'holidays'	type = 'contact' & group = 'friends'	1.000, 200, 5.000

Rule translation. In Cozy, documents are formatted in JSON with a set of key-values, potentially associated to a binary file. Simple implementations of *Filter* and *DI* check a set of filtering conditions and extract the *ITs* needed to subsequently identify the subject(s) based on the document key-value pairs. More complex implementations deal with the binary part of the document, e.g., image recognition or classification. *SI*, *IsS* and *MatchS* are implemented in PlugDB on the MiloDB [3] RDBMS. The S_{BR}, D_{BR}, G_{RR} and G_{except} data structures used to materialize ACL^{+}, ACL^{-} and $ACL^{?}$ are mapped in MiloDB relational tables. Tables *BRD*(*Rid int*, *Did char*(*32*), *A char*(*1*)) and *BRS*(*Rid int*, *Sid int*) materialize the union of S_{BR} and D_{BR} for all basic rules *BR*, with *Rid* the identifier of a basic rule granting authorization *A* on document *Did* (ids in Cozy are 32 bytes) to subject *Sid*. Tables *RR* and *Except*, of schema (*Sid int*, *Did char*(*32*), *A char*(*1*)), respectively materialize G_{RR} and G_{Except}. The set of $ACL^{?}$ are stored in 3 tables *BRD?*, *BRS?* and *RR?* with the same schema as respectively *BRD*, *BRS* and *RR*. The subjects and their identification traits are stored in table *SIT* (*Sid int*, *IT varchar*). *SI*, *IsS*, *MatchS*, as well as *Allowed* and *Who*, *What*, *Which* can trivially be implemented as SQL queries on these tables.

Initialization. At creation time, each sharing rule must be evaluated over all the documents of the personal cloud. This generates the environment costs shown in Fig. 3(left) for each document plus the time to evaluate the *Filters* of the rule and the remaining part of the rule evaluation tree if the document is qualified. To evaluate this cost, we generate documents with an average size of 1 KB which is typical for Cozy documents with no binary part. We assume large binary files don't impact the overall cost as the *Filters* first check the metadata and rule out the files without the expected type. Fig.3(middle) plots the time spent to run the initialization process for each rule of Table 1, depending on the number of considered documents. Conclusions are as follows: (i) the initialization cost of rule creation is acceptable; (ii) the environmental cost represents half of the total cost and is mainly due to communications between SEE, IE and UE which could largely be saved with other settings (e.g., Trustzone or SGX); and (iii) apart from environment costs, the cost of evaluating *Filters* is pre-

dominant because of the number of iterations (1 evaluation per document and rule) but not because of the cost of an elementary filter.

Maintenance. Each time a document *d* is inserted, the *Filters* of all rules are evaluated against *d* to check whether new ACLs can be produced. This is not an issue for basic rules, but may lead to a rescan of the personal cloud for reflexive rules if pending reflexive associations between subjects and documents are not maintained. To this end, we create table *RD*(*Rid int*, *Did char*(*32*), *IT varchar*) to record the rules and doc id pairs along with any identification trait which did not match an existing subject at the current time. Conversely, table *HR*(*Rid int*, *Sid int*) records the subjects ids qualified by that reflexive rule in order to detect an association with a future inserted document referencing one of these subjects. Given the shape of the operator trees (Fig. 1) the maintenance cost is determined by (1) the *Filters* evaluation for all active rules and (2) the cost of *IsS* when the document is qualified by *Filter*. Indeed, *IsS* cost depends on the cardinality of *S* and on the number of *ITs S* holds. Fig. 3(right) indicates the number of subjects and *ITs* per subject which remains compatible with the insertion of a document in less than 1s for a given number of rules. To vary the number of rules, we generate new rules having the same characteristics as the ones shown in Table 1. The maintenance cost for *BR* rules (not reported) always remains under 2.5ms per document, as it is independent of the number of subjects. Given Fig. 3(right), there is no performance issue linked to ACL maintenance (with 100 *RR* rules and without resorting on indexes, we can manage more than 1.000 subjects with 7 ITs for a maintenance cost of less than 1s per inserted document.

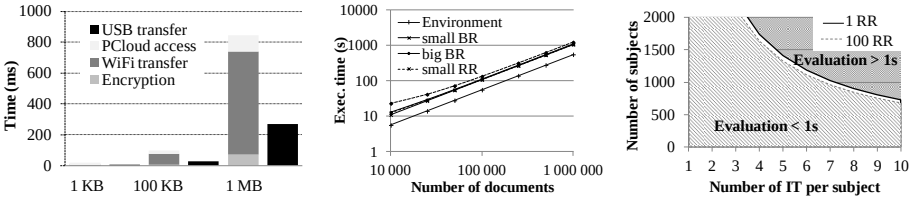

Fig. 3. Environmental costs for one document (left curve), rule initialization costs (middle) and rule maintenance costs in function of the subjects cardinality (right).

7 Conclusion

This paper introduces a new sharing paradigm for the personal cloud, called SWYSWYK (*Share What You See with Who You Know*), empowering individuals with new means to regulate by themselves the dissemination of their data with tangible enforcement guarantees. The model allows the expression of intuitive reflexive rules and lets the owner visualizes the net effects of these rules. Subjects are self-administered by design at rule translation and document insertion. We proposed an architecture to enforce the model and shown its feasibility through a performance evaluation performed on a secure DB engine (PlugDB) linked to an existing personal cloud platform (Cozy). While the personal cloud paradigm is pushed by recent legislation and smart disclosure initiatives, finding new ways to intuitively and securely

share personal data is paramount. We hope that this work actively contributes to this challenge.

References

1. Regulation (EU) 2016/679 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data. 27 April 2016.
2. Bertino, E., Ghinita, G., and Kamra, A. (2011). Access control for databases: Concepts and systems. Foundations and Trends in Databases, 3(1-2).
3. Brickley, D., and Miller, L. FOAF vocabulary specification 0.91. TR ILRT Bristol, 2007.
4. Fang, L., LeFevre, K. Privacy wizards for social networking sites. In ACM WWW, 2010.
5. Squicciarini, A.C., Sundareswaran, et al. A3P: adaptive policy prediction for shared images over popular content sharing sites. In ACM Hypertext and hypermedia (HT), 2011.
6. Tootoonchian, A., Saroiu, S., Ganjali, Y., and Wolman A. Lockr: better privacy for social networks. In Conf. Emerging networking experiments and technologies (CoNEXT), 2009.
7. Van Kleek, M., Smith, D.A., Shadbolt, N., and schraefel, m.c. A decentralized architecture for consolidating personal information ecosystems: The WebBox. In PIM, 2012.
8. Seong, S-W., Seo, J., Nasielski, M., Sengupta, D., et al. PrPl: a decentralized social networking infrastructure. In ACM Mobile Cloud Computing & Services (MCS), 2010.
9. Ali, M., et al. (2015). SeDaSC: secure data sharing in clouds. IEEE Systems Journal.
10. Thilakanathan, D., Chen, S., Nepal, S., and Calvo, R.A. Secure Data Sharing in the Cloud. In Security, Privacy and Trust in Cloud Systems, 2014.
11. Wang, F., et al. Cryptographically Enforced Access Control for User Data in Untrusted Clouds. In USENIX Symposium on Networked Syst. Design and Implem. (NSDI), 2016.
12. Baden, R., Bender, A., Spring, N., et al. Persona: an online social network with user-defined privacy. ACM SIGCOMM Comp. Com. Review, 39(4), 2009.
13. Guha, S., Tang, K., and Francis, P. NOYB: Privacy in online social networks. In ACM workshop on Online Social Net., 2008.
14. Yuan, L., et al. Privacy-preserving photo sharing based on a secure JPEG. In CCC, 2015.
15. Liu, Y., Gummadi, K. P., Krishnamurthy, B., and Mislove, A. Analyzing facebook privacy settings: user expectations vs. reality. In ACM SIGCOMM, 2011.
16. Mazurek, M.L., Liang, Y., et al. Toward strong, usable access control for shared distributed data. In USENIX conference on File and Storage Technologies (FAST), 2014.
17. Wang L., Wijesekera D., and Jajodia S. A Logic-based Framework for Attribute based Access Control. In ACM workshop on Formal methods in security engineering (FMSE), 2004
18. Geambasu, R., Balazinska, M., Gribble, S.D., and Levy, H.M. Homeviews: peer-to-peer middleware for personal data sharing applications. In ACM SIGMOD, 2007.
19. Carminati, B., Ferrari, E., and Perego, A. Rule-Based Access Control for Social Networks. In On the Move to Meaningful Internet Systems, 2006.
20. Mori, J., Sugiyama, T., and Matsuo, Y. Real-world oriented information sharing using social networks. In ACM SIGGROUP (GROUP), 2005.
21. Park, J., Sandhu, R. (2004). The UCON ABC usage control model. In ACM TISSEC, 7(1).
22. Anciaux, N., Bouganim, L., Pucheral, P., Guo, Y., Le Folgoc, L., and Yin, S. MILo-DB: a personal, secure and portable database machine. In DAPD, 32(1), 2014.
23. Anciaux, N., Lallali, S., Popa, I. S., and Pucheral, P. A scalable search engine for mass storage smart objects. In PVLDB, 8(9), 2015.
24. Costan, V., and Devadas, S. Intel SGX Explained. IACR Cryptology ePrint Archive, 2016.
25. Alves, T., and Felton, D. Trustzone: Integrated hardware and software security. ARM white paper, 3(4), 2004.