SWYSWYK: A Privacy-by-Design Paradigm for Personal Information Management Systems

Paul Tran-Van, Nicolas Anciaux, Philippe Pucheral

SWYSWYK: a Privacy-by-Design Paradigm for Personal Information Management Systems

Paul Tran-Van[1,2,3]
[1] Cozy Cloud
158 rue de Verdun
92800, Puteaux, France
paul@cozycloud.cc

Nicolas Anciaux [2,3]
[2] Inria Saclay-Île-de-France
1 rue d'Estienne d'Orves
91120 Palaiseau, France
nicolas.anciaux@inria.fr

Philippe Pucheral [2,3]
[3] DAVID Lab., University of Versailles
45, Av. Etats-Unis,
78035 Versailles, France
philippe.pucheral@uvsq.fr

Abstract

Pushed by recent legislation and smart disclosure initiatives, Personal Information Management Systems (PIMS) 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. Indeed, existing sharing models are difficult to administrate and securing their implementation in user's computing environment is an unresolved challenge. This paper advocates the definition of a Privacy-by-Design sharing paradigm, called SWYSWYK (*Share What You See with Who You Know*), dedicated to the PIMS context. This paradigm allows each user to physically visualize the net effects of sharing rules on her PIMS and automatically provides tangible guarantees about the enforcement of the defined sharing policies. Finally, we demonstrate the practicality of the approach through a performance evaluation conducted on a real PIMS platform.

Keywords: Personal Information Management Systems (PIMS), Privacy-by-Design, Access control, data security

1. Introduction

We are witnessing an exponential accumulation of personal data on central servers, gathered by administrations and companies, or created by individuals and replicated in the cloud for convenience. Centralization exacerbates the risk of privacy leakage due to piracy[1], scrutinization and opaque business practices. Today, a rebalancing of personal data management is occurring worldwide. *Smart disclosure* initiatives are pushed by legislators (e.g., EU General Data Protection Regulation [17]) and industry-led consortiums (e.g., blue button for medical records and green button for electricity in the US, Midata in the UK, MesInfos in France). Smart disclosure enables individuals to get back their personal data from companies or administrations that collected them. Concurrently, the *Personal Information Management System* (PIMS) paradigm has been conceptualized [1], [<removed for refereeing>], and emerges in the commercial sphere (e.g., Cozy Cloud, ownCloud, SeaFile). PIMS 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.

But this gravity shift of data management from organizations to individuals raises new fundamental issues. Empowering citizens to leverage their personal data leaves the privacy and security issues in user's hands, a paradox if we consider the weaknesses of individuals' defenses in terms of computer security and ability to administer sharing policies. Existing sharing models

[1] Yahoo recent hack attack is emblematic from this phenomenon, as mentioned in "Yahoo state hackers stole data from 500 million users" - BBC News. www.bbc.co.uk/news/world-us-canada-37447016

(e.g., DAC, RBAC, MAC [6]) are geared towards central authorities and their secure enforcement requires a deep expertise, out of reach of individuals. Conversely, decentralized tools (e.g., PGP, Web of Trust models [23] or FOAF dissemination rules [7]) put on individuals the burden of defining manually each basic sharing rule and leave them on their own to manage error-prone cryptographic protection against piracy. Hence, without appropriate answers, the risk is high to see individuals delegate the administration of their PIMS to centralized service providers. The circle would come back around, with service providers now in possession of the complete individual's digital history.

The objective of this paper is precisely to address this issue. We do not suggest yet another access control model. Rather, we propose a generic paradigm contributing to the call for Privacy-by-Design principle when designing any PIMS platform. More precisely, this paper makes the following contributions:

- It proposes a new paradigm called SWYSWYK (*Share What You See with Who You Know*) helping the PIMS owner to visually check and sanitize the net effect of a sharing policy over her data, whatever the access control model used to generate this policy.
- It defines a reference architecture supporting the SWYSWYK paradigm which provides to the PIMS owner tangible guarantees about the enforcement of this policy.
- It conducts a thorough performance evaluation over an instance of this reference architecture, combining an existing PIMS platform with a tamper resistant hardware device to demonstrate the practicality of the approach.

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

2. Related works and problem formulation

2.1. Access Control and sharing models

To avoid any confusion between users, let us call *owner*, the owner of a PIMS and *subjects*, the users the PIMS owner wants to interact and exchange data with. Access control models like DAC, RBAC, MAC, ABAC or TBAC [6] are widely supported in the centralized database context. They unfortunately share the following characteristics: (1) the access control administration is a complex and critical task usually handled by security experts; (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. This makes these models badly adapted to a personal usage in the PIMS context, except if the PIMS comes with predefined policies that the PIMS owner simply accepts or denies. In the latter case, this contradicts *user's empowerment*, a foundational principle of Cavoukian's Privacy-by-Design principles [8] and of the new GDPR regulation [17].

Conversely, flexible sharing models have been designed to cope with the decentralized nature of the Web. In this respect, they seem better adapted to the PIMS context. For instance, Web of Trust-like models allow subjects to authenticate thanks to their public key and are identified through public profiles [20], to which access rights are associated and defined by the owner. However, it requires the owner to manually assign the authorizations, for each subject and for each object, which can quickly become tedious and error-prone. To ease data sharing administration, [15] implements a distributed tag-based file-system where an owner can grant access to her tagged files using a logic-based access control. [11] defines an SQL-based language to let applications create queryable views on the shared data and transmit capabilities to granted subjects. In [7], subjects are granted access depending on their FOAF relationship properties, such as type, graph depth and a computed trust value. [5] tries to federate social identities and to classify it in three types (awareness, symmetry and mutual acceptance), each of them corresponding to a set of predefined, yet customizable policies. Despite all these efforts, the cognitive load on the owner is such that it often leads to consider data sharing as an intractable burden, letting desperate owners define far too permissive policies [14].

2.2. Data security and data sharing enforcement

A corollary of data sharing definition is the enforcement of the corresponding policies (i.e., data to be exchanged) as well as the protection of the PIMS itself (i.e., data at rest) against confidentiality attacks. The way PIMS can be secured depends on the data hosting model. Traditional solutions (e.g., Google Drive, Microsoft OneDrive, Apple iCloud) delegate all the control to cloud providers, thereby definitely hurting user's empowerment. [2], [19], [21] focus on data encryption in untrusted clouds, while [4] presents a decentralized alternative to social networks based on ABE encryption. [12], [22] use obfuscation schemes, where data is substituted for the former and scrambled for the latter. Sharing policies are implemented here by means of encryption but, this requires the owner to manually define who can access which data on a case-by-case basis. Moreover, the security of the cryptographic keys is either delegated to the server or left in owner's hand. Finally, self-hosting solutions (e.g., ownCloud, Lima, CloudLocker ...) let owners install their own PIMS at home, under their control. In all approaches, either the owner must give up the control over her data or she inherits the task of administering and protecting herself her PIMS, a task that she cannot properly assume.

2.3. Problem formulation

Hence, data sharing and data security can be tackled in many ways, but none of the previously work won the battle for adoption. The difficulty comes from two contradictory facts. On one hand, the owner is de facto the PIMS administrator while it is illusory to expect her gaining expertise to secure her PIMS against all forms of attacks or to use tricky protocols to exchange cryptographic secrets with partners. On the other hand, delegating these tasks to a PIMS provider is in frontal opposition with the foundational user's empowerment principle. We can derive from this statement two major properties to be met in the quest of a real Privacy-by-Design PIMS:

- *Enlighten empowerment:* the effects of all owner’s decisions must be perceivable and understandable by herself. In other words, whatever the sharing model used to express a sharing policy, the owner may be in capacity to visualize the net effects of these rules on her PIMS and to easily sanitize them if required.
- *Tangible enforcement:* the logic of the reference monitor enforcing the sharing policy must itself be understandable by the owner and the platform implementing this logic must be trusted by her. Making the reference monitor logic understandable leads to an extreme simplicity, with the side effect to enable running it on a tamper-resistant personal device kept in user's hand, thereby reinforcing the trust in the platform.

3. SWYSWYK paradigm

3.1. SWYSWYK baseline

As said above, our objective is not to introduce yet another access control model. Rather, we do the assumption that the access control policy is defined by the genuine access control model provided by the PIMS platform. Hence, it could be based on any of the model mentioned in Section 2.1, with the few restrictions mentioned below.

First, we do the assumption that the sharing granularity is the document and that every shareable document is made viewable by the PIMS owner. In other words, this means that there exist a Viewer application trusted by the owner which delivers an interpretable view of each type of document to be shared (e.g., potentially transforms a binary format into a text, an image or a graphic). 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). Second, each subject the owner wants to interact with should correspond to a PIMS viewable document as well (e.g., a contact record, a resume, a picture). Third, the access control policy is materialized by a set of ACL (*Access Control List*, also called *permissions* hereafter) of the form < *s*, *d*, *a* >, where *s* and *d* respectively refer to a subject and a document stored in the PIMS, and *a* is an action granted to *s* on *d*. The PIMS owner may have the ability to check these ACLs and freely filter out whose which presumably hurt her privacy. The owner is not expected to check all ACLs but simply to validate some suspicious ones detected as such by administration tools detailed next.

Materializing all ACLs, making them viewable by the owner and letting her filter out compromising ones give substance to the *Share What You See with Who You Know* (SWYSWYK) principle. This is in frontal opposition with traditional approaches where sharing policies are defined by a set of - potentially complex - rules, evaluated on the fly by the reference monitor to grant or deny accesses to documents. Indeed, we don't believe that a lambda individual can figure out the result of a powerful solver taking as input a set of potentially conflicting positive/negative sharing rules. Traditional approaches may actually contribute to the opacity of the PIMS access control management, hurt user's empowerment and lead to an opposite effect of that expected, that is defining too permissive policies or delegating the control to a third party. Conversely, the trust assumption made in SWYSWYK can be summarized by the following motto: *do not blindly trust sharing rules but trust your reference monitor to let you examine and adjust the produced permissions and enforce data dissemination accordingly*. Hence, in contrast to rule-based reference monitors, the logic of a SWYSWYK reference monitor can be trivially understood by anyone, that is to say *operation a on d is granted to s iff* $(s,d,a) \in ACL$. This logic contributes itself to the *enlighten empowerment* property. Note that adding such user's validation and sanitization process does not compromise the soundness of the genuine PIMS sharing model. Indeed, the model remains *consistent* by construction (the decision is unique), *complete* (the decision always exists) and can finally be evaluated in *logarithmic time* over the sanitized materialized set of ACLs.

The tricky point of the SWYSWYK paradigm lies in the detection and validation of suspicious ACLs. The global ACL validation process works as follows. First, the genuine sharing policy is translated into a materialized set of candidate ACL named ACL^*. Second, suspicious ACLs are detected and put in quarantine, in a set named $ACL^?$, waiting for the decision of the PIMS owner. Non suspicious ACLs are directly integrated in the set ACL^+, the unique set to be considered by the reference monitor to grant or deny accesses to documents. Third, the PIMS owner sanitizes the set of suspicious ACLs on a case-by-case basis. She visualizes the net effect of suspicious ACLs in $ACL^?$ and decides either to store them in ACL^+ if she considers them innocuous or in ACL^- otherwise. The objective of materializing ACL^- is nothing but avoiding storing in $ACL^?$ ACLs for which a negative decision has already been made in the past, hence avoiding unnecessary PIMS owner intervention.

We propose two mechanisms to automatically detect suspicious ACLs and feed $ACL^?$ from the content of ACL^*. The first mechanism is based on an Advisor process identifying elements of ACL^* which are contradictory to past decisions (i.e., similar to ACLs previously classified in ACL^-). This mechanism is based on the assumption that owners exhibit a rather stable data disclosure behavior over time, as already demonstrated with the usage of emails [18]. The second mechanism is user-defined triggers highlighting ACLs for which special care should be taken, because of the sensitiveness attached to the documents and/or to the subjects involved in these ACLs. These two complementary mechanisms are detailed below.

3.2. Suspicion based on past decisions

Given any candidate ACL $(s,d,a) \in ACL^*$, the goal is to suggest a decision (i.e., *accept* or *suspect*) to the owner. We denote by $r_{(s,d,a)}$ this decision. The decision being unique by construction, (s,d,a) is discarded if it already exists in ACL^+ or ACL^-. Otherwise, the Advisor process computes a distance between the candidate ACL (s,d,a) and the ACLs resolved in the past (i.e., elements of ACL^+ or ACL^-) and make a decision similar to the closest resolved ACL if the corresponding distance is below a given threshold.

The way distances are computed is an interesting open issue, which can integrate a comparison between attributes of documents and subjects as well as a comparison of histories of past decisions. Finding the best metric to compute this distance is let for future work. We simply validate the idea thanks to a basic metric, the benefit of which being to be computable in any resource constrained secure execution environment (e.g., secure chips).

Let S be the set of all subjects identified in the owner's PIMS and $S' \subseteq S$ be the set of all subjects s' different from s with a resolved decision concerning d and a, that is $S'=\{s' \in S, s' \neq s, \exists(s',d,a) \in ACL^+ \cup ACL^-\}$. Let h_s (resp. $h_{s'}$) denote the history of all resolved decisions regarding s (resp. s'), that is $h_s=\{(s,d,a,r_{(s,d,a)}), \exists(s,d,a) \in ACL^+ \cup ACL^-\}$. The $Resolve(s,d,a)$ algorithm presented below returns a value for $r_{(s,d,a)}$ based on the computation of a distance $Dist(h_{s'}, h_s)$ between h_s and all $h_{s'}$ and a confidence level for that value which helps the owner making her final decision. This algorithm is based on the assumption that, if s and s' share strong similarities in their sharing history with the owner, a past decision taken on (d,a) for s' is likely to apply for s as well. Hence, if Resolve suggests 'Accept', there is no reasonable reason to doubt the genuine sharing decision and the related ACL is stored in ACL^+. Otherwise, it falls in quarantine in $ACL^?$. A symmetric algorithm could be based on the assumption that, if d and d' share strong similarities in their sharing history by the owner, a past decision taken on (s,a) for d' is likely to apply for d. The adaptation of the algorithm below to this new assumption is direct.

```
**Algorithm 1.  Resolve**
**Input:**     $t = <s,d,a> \in ACL^*$
**Output:**    (r, p) with r a binary decision (Accept or Suspect)
               and $p \in [0,1]$ the confidence in that decision.
if $t \in ACL^+$ **return** ('Accept', 1);
if $t \in ACL^-$ **return** ('Suspect', 1);
/* build $h_s$ the history of decisions for subject s*/
$h_s$ = { (acl, 'Accept'), acl.s=t.s, acl $\in ACL^+$ }
    $\cup$ { (acl, 'Except'), acl.s=t.s, acl $\in ACL^-$ };
/* build H, the set of histories of other subjects than s with
decision a for document d */
H={$h_{s'}$, s'$\neq$s, $\exists$ acl$\in h_{s'}$, acl.d = t.d, acl.a = t.a };
$h_{close} \leftarrow$ the history with smallest Dist($h_s$, $h_{s'}$), $\forall$ $h_{s'} \in$H ;
r  $\leftarrow$ the decision for (d,a) in $h_{close}$;
p  $\leftarrow$ 1- Dist(h, $h_{close}$) ;
**return** (r, p);
```

3.3. Suspicion based on sensitiveness

An ACL can be considered suspicious either because it involves a sensitive subject (e.g., my manager), a sensitive document (e.g., a compromising picture or a part of my medical folder) or because the association between a particular subject and document 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). The sensitiveness of subjects, documents and associations is left to the PIMS owner appreciation. We provide watchdog triggers, which can be specified by the owner, to easily identify ACLs targeting sensitive subjects, sensitive objects or sensitive associations between them. To this end, we make the usual assumption that PIMS documents are linked to metadata (e.g., doctype, date, author, owner-defined tags, etc) and that metadata can be queried by a predicate-based language. Let $Q_E(S)$ denote the queries expressed over - metadata linked to - documents describing subjects and $Q_E(D)$ denote queries over - metadata linked to - shared documents, with E the predicate expression of the query. We propose three types of predefined watchdog triggers:

- $What'sNewforS(E,A) \rightarrow \{(s,\{(d,a)\}) \,/\, (s,d,a)\in ACL^{*} \wedge s\in Q_E(S) \wedge a=A\}$: 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 manager?").
- $Who'sNewforD(E,A) \rightarrow \{(d,\{(s,a)\}) \,/\, (s,d,a)\in ACL^{?} \wedge d\in Q_E(D) \wedge a=A\}$: identifies, for each selected (sensitive) document *d*, the new set of subjects *s* that are granted to perform action *a* on them (e.g., "which new subjects have a read access to my medical records?").
- $WhichNewSD(E,E',A) \rightarrow \{(s,d,a) \,/\, (s,d,a)\in ACL^{*} \wedge s\in Q_E(S) \wedge d\in Q_{E'}(D) \wedge a=A\}$: identifies new ACLs combining a selection of (sensitive) subjects and documents (e.g., "which new authorizations my colleagues have on my family photos?").

4. SWYSWYK reference architecture

This section introduces a reference architecture implementing the SWYSWYK paradigm while providing the *tangible enforcement* property, thereby making the contribution complete. The objective of this architecture is to protect the owner's data by construction, without requiring any action from the owner, apart from the validation of suspicious ACLs. Back to Cavoukian's Privacy-by-Design principles [8], this means contributing to the *Privacy-by-Default* and *End-to-End security* principles. This architecture, presented in Figure 1, distinguishes three main parts with different assumptions in terms of trustworthiness: (i) an *untrusted environment (UE)* on which no security assumption is made for the code nor for the data, (ii) an *isolated environment (IE)* on which general purpose code can be run with the guarantee that it cannot leak any information but with no guarantee about the soundness and honesty of its output (i.e., code can be corrupted) and (iii) a *Secure Execution Environment (SEE)* which runs only certified core programs and protects data and code against snooping and tampering. We postpone to Section 5 concrete illustrations of *UE*, *IE* and *SEE* environments, considering here these security assumptions as given. That said, the components of the architecture are as follows.

PIMS data system. Our objective is to adapt to any existing PIMS platform. Thus, no assumption can be made on the intrinsic security of such platform. It is then part of the untrusted environment *UE*. All documents of the PIMS need then be stored encrypted in this area to protect them against confidentiality attacks.

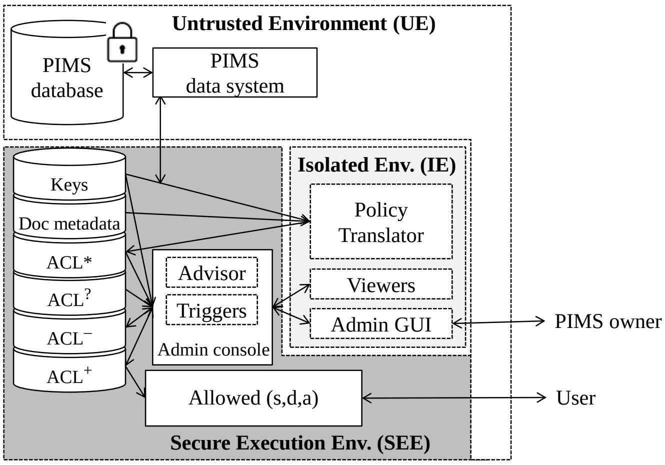


Fig. 1. SWYSWYK Reference Architecture

Reference monitor. The reference monitor is part of the secure core of the architecture. It must be embedded into the *SEE* to guarantee that it cannot be bypassed, observed nor corrupted by any external application. Roughly speaking, the reference monitor evaluates *Allowed*(*s*,*d*,*a*) requests and delivers *true* iff ($< s, d, a > \in ACL^{+}$) and *false* otherwise. It acts as an incorruptible doorkeeper for the whole PIMS data system. Whenever *Allowed*(*s*,*d*,*a*) is evaluated to *true*, document *d* is decrypted in the *SEE* before being delivered to subject *s*. Given the simplicity of the *allowed* function, it can be hosted in many kinds of (tamper-resistant) *SEE* kept in user's hand (e.g., SIM cards in users' smartphones, secure personal tokens [<removed for refereeing>], TPM).

Policy translator. The policy translator is in charge of running the rules of the genuine sharing policy over the PIMS documents and of materializing candidate ACLs in *ACL**. The policy translator must run in the isolated environment *IE*. Indeed, it cannot be part of the *UE* because some sharing rules may depend on the documents content and then require to decrypt these documents. It cannot be part of *SEE* either without hurting the genericity requirement. Indeed, the complexity of the policy translator is linked to the expressive power of the genuine sharing model and running complex code in a *SEE* can be highly challenging. Note that running the policy translator in *IE* does not compromise security since no data can leak from *IE* by construction. This property is guaranteed by running each code in *IE* in an *isolated container*[2] with restricted permissions. Typically, the policy translator runs inside a container with *read access* to the PIMS documents, *write access* to ACL^* in *SEE* and no additional privilege (typically, no network access). Hence, the policy translator cannot leak anything and cannot be observed by concurrent processes (by definition of an isolated container).

Administration console. The administration console is used to help the PIMS owner to perform the ACL validation task. As detailed in Section 3, this console runs the Advisor process and watchdog triggers over ACL^* and puts suspicious ACLs in quarantine in $ACL^?$. The administration console must be trusted, but cannot be entirely executed inside the *SEE*. Indeed, it involves interactions with the owner through a GUI and requires displaying the content of documents and subjects. Thus, the *Administration GUI* and *document and subject viewers* must run in isolated containers in *IE* to prevent any information leakage.

Internal data structures. The ACL^*, $ACL^?$, ACL^+ and ACL^- sets and the document metadata must all be stored inside *SEE* for obvious security reasons. Storing them in *UE* would incur prohibitive decryption and integrity checking costs during the evaluation of *Allowed*, *Advisor* and *watchdog triggers*. However, the counterpart of the reference monitor simplicity is the potential cost of evaluating *Allowed*(s,d,a) in the case of extremely large ACL^+. This situation may happen whenever a rule of the genuine sharing policy associates a large set of subjects with a large set of documents. If storing millions of ACLs and indexing them on both s and d is not a challenge in a standard setting, it may become intractable in more constrained secure hardware environments (e.g., smartcards, TPM, quantified-self devices, smart watches). Such combinatorial explosion can be easily avoided thanks to a compact representation. Each ACL set actually materializes a bipartite graph $G = (S, D, A)$ where the sets of vertices S and D respectively represent the set of subjects and documents and the edges A represent the authorizations linking them. Each sharing rule R defines a complete bipartite subgraph G_R of G. As such, each G_R can be stored in a compressed form as a set $S_R \subseteq S$ of subject vertices and a set $D_R \subseteq D$ of document vertices, A_R being implicit.

5. SWYSWYK validation

We validate the proposed solution on a concrete instance of the reference architecture introduced in Section 4, generic enough to draw conclusions for other potential targeted instances. We show its efficiency both qualitatively and quantitatively on real and synthetic datasets.

[2] 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 [3] or SGX processors [9].

5.1. Experimental Platform

The experimental platform used to validate our proposal is based on a combination of the Cozy system[3] and PlugDB[4]. Cozy is a representative open-source PIMS suite, gathering personal data from multiple sources and organizing it in a document style database (using CouchDB[5]). Our solution is actually independent of the PIMS platform and other PIMS could have been used as well (e.g., ownCloud, Sandstorm, Databox [16], etc). PlugDB is a secure and open hardware/software platform. It combines a smartcard to store cryptographic secrets, a microcontroller (MCU) running a relational database engine queried in SQL which can efficiently manage a large database, and a microSD flash card storing the database with crypto-protection against snooping and tampering. We used a simple genuine sharing model, able to share a selection of documents with a selection of subjects (e.g., “share the Holidays photo album with my family group”), where rules are produced by Cozy apps and evaluated in the policy translator.

This experimental platform, presented in Figure 2, is an instance of the reference architecture: (i) Cozy runs on a personal computer linked to the network (Internet access) and represents the *untrusted environment (UE)*; (ii) the policy translator, the administration GUI and the viewers are installed on a Raspberry Pi without any network connection which represents the *isolated environment (IE)*; and (iii) the reference monitor, the Advisor and the watchdog triggers are installed within PlugDB, which represents the *SEE* and communicates with both the personal computer (in WiFi IEEE 802.11n) and the Raspberry Pi (in High Speed USB 2.0). Validating the approach in such highly constrained environment is a proof of its simplicity and of the possibility to formally prove its code. It also demonstrates the ability to embed a SWYSWYK engine in any kind of constrained *SEE*, including IoT objects.

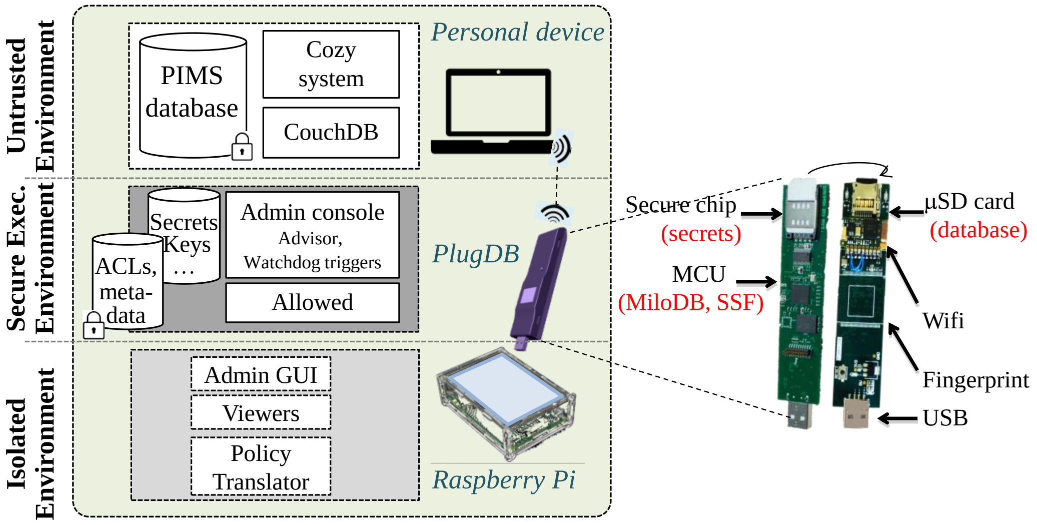


Fig. 2. Experiment platform: soft (left) & hard (right).

In this platform, the isolation property is "physically" guaranteed by the Raspberry Pi, such that its enforcement cannot be questioned. Others target architectures can be envisioned. For example, a certified hypervisor running on top of Intel SGX on the personal computer itself can provide a logical implementation of *IE* (Figure 3, left part). Smart devices equipped with a SIM card, a flash memory card and an ARM Trustzone processor - as many smartphones and tablets today - are other options (Figure 3, right part). The major difference between these alternatives is on the way *UE*, *IE* and *SEE* communicate. In this section, we measure and isolate the communication cost to allow drawing general conclusions.

[3] https://cozy.io/en/
[4] https://project.inria.fr/plugdb/en/
[5] http://couchdb.apache.org/

In our experimentations, the personal device implementing *UE* has a 3GHz Intel Xeon E5-1660 CPU, 8 GB of RAM and a 500 GB 10.000 RPM hard drive. The *SEE* uses a STM32F417GH6 MCU, which embeds a 168 MHz ARM Cortex M4 CPU, 192 KB of RAM and 1 MB of NOR storage. The *IE* is a Raspberry Pi 3 with 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*.

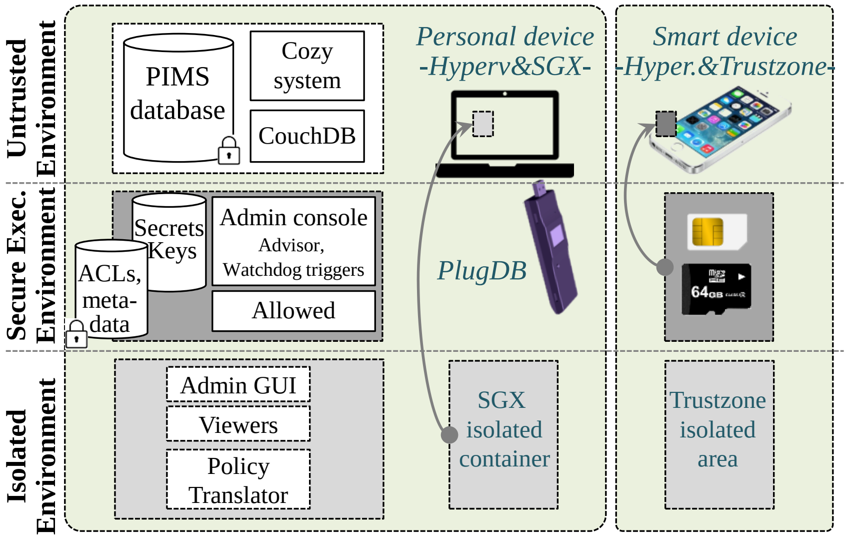
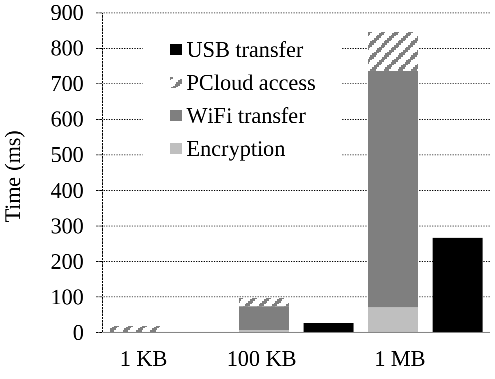


Fig. 3. Other examples of targeted architectures.

5.2. Environmental Costs

We first measure the environmental cost when inserting a new document in the PIMS. Figure 4 pictures the four components of this cost: (1) transfer costs (WiFi) of the document between *UE* and *SEE* (in cleartext from *UE* to *SEE* and in encrypted form back), (2) document encryption cost by *SEE*, (3) insertion of the encrypted document into the Cozy system and (4) transfer cost (USB) of the cleartext document from *SEE* to *IE* (where the policy translator checks whether this document matches some sharing rules and sends back ACLs in the positive case). Note that step 4 can be performed in parallel with the first three steps, explaining why we isolate its cost (USB transfer) in the figure.



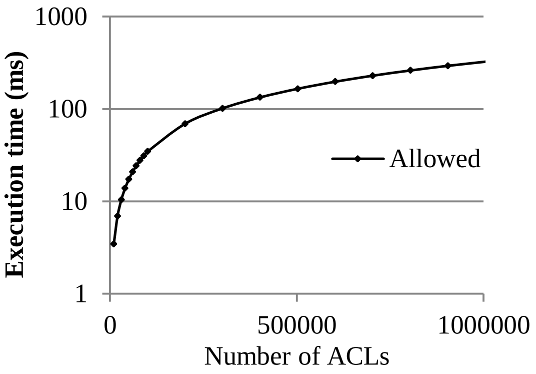


Fig. 4. Environmental costs for one document. **Fig. 5.** Execution time of *Allowed* in *SEE*.

The environmental costs are negligible for small size documents, whereas the WiFi transfer dominates for large documents. Note that the communication costs could be significantly improved by using newer WiFi and USB norms (WiFi 802.11ac and USB 3.1 offer data rates theoretically 2.9 and 20.8 times faster than 802.11n and USB 2.0, respectively). Note also that these costs are mainly linked to the way the PIMS security is implemented rather than being linked to the SWYSWYK model on its own. Hence, alternative architectures such as those pictured in Figure 3 would greatly reduce these costs.

5.3. Reference Monitor cost

The *Allowed* function takes as input a triple (*s*, *d*, *a*) and returns *true* if subject *s* is granted authorization *a* on document *d* and *false* otherwise. As explained in Section 4, executing this function sums up to check the existence of the triplet (*s*, *d*, *a*) in ACL^+ which is represented in a compact way by a complete bipartite graph for each rule. This leads to store the ACLs in relational tables of the form *Rule(RuleId, A)*, *NodeS(RuleId, SubjectId)*, *NodeD(RuleId, DocId)*, so that *Allowed* is a simple SQL query on these tables. Selection indexes have been built in PlugDB to speed up this query.

Figure 5 gives the execution time of *Allowed*, increasing the number of ACLs up to 1 million. The figure shows that the performance of *Allowed* remains acceptable (i.e., below one second) in all cases up to 1 million of ACLs, while such a case is very unlikely. This demonstrates the effectiveness of the approach.

5.4. Qualitative analysis of ACL validation

We evaluate here the quality of the *Resolve* function by applying it on a real dataset. As we are not aware of any public ACL dataset, we chose personal emails datasets for this qualitative analysis. Indeed, famous email datasets have been made available to the community and emails reflect in some respect the policy of a user in terms of personal documents dissemination (and of other types of data sharing scenarios [18]). For the sake of representativeness, we use four different real datasets: the *ENRON* dataset [10], the *Hillary Clinton*'s emails revealed by Wikileaks [13] and the emails of one author of this paper (called *people1*) and of one of his family member (called *people2)*.

In this experiment, we consider the email body as a personal document d_i granted by the sender to the set of recipients $\{s_j\}$. ACL^+ is then given by the set $\{(s_j, d_i, \text{'}read\text{'})\}$. To generate ACL^*, we randomly choose some 'legitimate' (i.e., existing) elements in ACL^+ and some 'illegitimate' elements built by combining existing emails *d* in $\{d_i\}$ with existing recipients *s* in $\{s_j\}$ which are not recipients of *d*. We expect the *Resolve* function to *accept* the legitimate elements of ACL^* and identify the others as suspicious and store them in $ACL^?$.

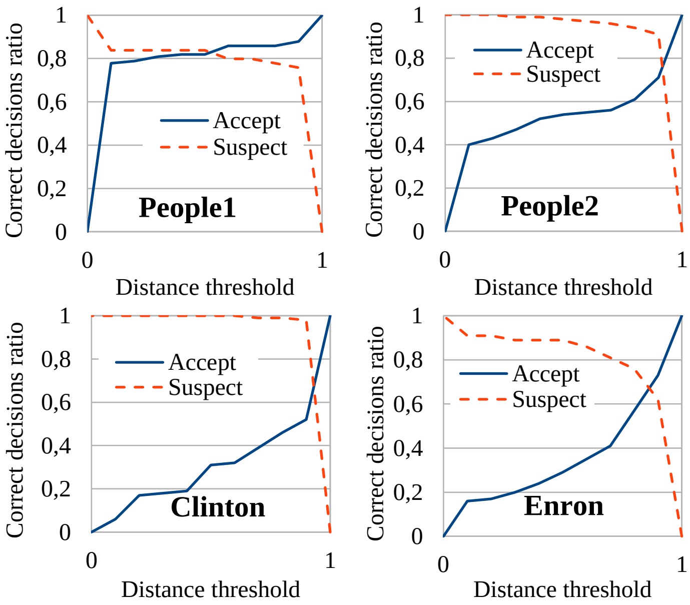


Fig. 6. Resolve function success rate for ACL^+ and $ACL^?$

The function that we use to evaluate the distance between the histories of two subjects sums up to count the number of emails where the two subjects appear as recipients, and apply the reciprocal function. Such a simple *Dist* function can be easily implemented in SQL on top of PlugDB by using a count query. The evaluation shows that such a simple function makes sense and allows to get a *Resolve* implementation close to Algorithm 1. For each element (d_i, s_j) of ACL^*, we compute the distance between s_j and all the others recipients of this email (i.e., the subjects who are granted access to d_i in ACL^+) and keep the subject with the smallest distance. If the distance is below a minimum threshold *t*, *Resolve* accepts the ACL and puts it in ACL^+ or puts it in quarantine in $ACL^?$ otherwise.

In Figures 6, we generate 50 elements in ACL^* half of which being legitimate. The process was repeated 1.000 times and the figures plot the mean of all the runs. The results are provided for each dataset, varying the distance threshold *t*. The lower *t*, the closer an element of ACL^* needs to be from an element of ACL^+ to be suggested by *Resolve* as *Accept*. This explains the respective shapes of the curves Accept and Suspect which plot the success rate of an $ACL^?$ to be correctly categorized into ACL^+ or $ACL^?$. The optimal value for *t* corresponds to the crossing point of both curves, where the success rate is around 80% for both Accept and Suspect classification. Optimal *t* depends on each user behavior, calling for a training process. Typically, optimal *t* for *Clinton* is high because *Clinton* address book is extremely large, resulting in a wide dispersion in the recipient distribution. Conversely, *People1* is more intimate in his behavior leading to a high concentration of the recipients. To further improve the success rate, others variables could be integrated in the decision process, such as the recency of the history entries [18] or their semantic proximity. Anyway, this evaluation shows that even a simple decision process makes sense and is compatible with a highly constrained secure execution environment.

6. Conclusion

This paper introduces a Privacy-by-Design sharing paradigm, called SWYSWYK (*Share What You See with Who You Know*), dedicated to the PIMS context. This paradigm allows each owner to visualize the net effects of sharing rules on her PIMS and to easily sanitize the sharing policy. Moreover, it provides tangible guarantees about the enforcement of the sanitized sharing policies without user intervention. Finally, we have shown the effectiveness of the approach through a performance evaluation performed on a secure DB engine (PlugDB) linked to an existing PIMS platform (Cozy).

Many exciting open issues remains to be investigated. A first perspective is linked to the security analysis of SWYSWYK which should be conducted for other architectures, e.g., based on processors like [3], [9] offering new forms of secure and isolated execution environments. A second important issue is the definition of more powerful tools helping the owner to detect suspicious ACLs. While the PIMS paradigm is pushed by recent legislation and smart disclosure initiatives, finding new ways to intuitively and securely share personal data is paramount. Otherwise, the risk is high to see desperate individuals delegate the administration of their PIMS to centralized service providers, a heresy in a period where stronger user's empowerment is called for. We hope that this work actively contributes to this challenge.

References

1. Abiteboul, S., André, B., Kaplan, D., Managing your digital life. CACM, 58(5), 2015.
2. Ali, M., Dhamotharan, R., Khan, E., Khan, S. U., Vasilakos, A. V., Li, K., and Zomaya, A. Y. (2015). SeDaSC: secure data sharing in clouds. IEEE Systems Journal.
3. Alves, T., and Felton, D. Trustzone: Integrated hardware and software security. ARM white paper, 3(4), 2004.
4. Baden, R., Bender, A., Spring, N., Bhattacharjee, B., and Starin, D. Persona: an online social network with user-defined privacy. In *ACM SIGCOMM Computer Communication Review*, *39*(4), 2009.

5. Bellavista, P., Giannelli, C., Iannario, L., Goix, L-W., and Venezia, C. Peer-to-Peer Content Sharing Based on Social Identities and Relationships. In IEEE Internet Computing, 18(3), 2013.
6. Bertino, E., Ghinita, G., and Kamra, A. (2011). Access control for databases: Concepts and systems. Foundations and Trends in Databases, 3(1-2).
7. Carminati, B., Ferrari, E., and Perego, A. Rule-Based Access Control for Social Networks. In On the Move to Meaningful Internet Systems, 2006.
8. Cavoukian, « Privacy by design - The 7 foundational Principles » https://www.ipc.on.ca/images/Resources/7foundationalprinciples.pdf
9. Costan, V., and Devadas, S. Intel SGX Explained. IACR Cryptology ePrint Archive, 2016.
10. Enron dataset, available at https://www.cs.cmu.edu/~enron/
11. 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.
12. Guha, S., Tang, K., and Francis, P. NOYB: Privacy in online social networks. *ACM workshop on Online Social Net.*, 2008.
13. H. Clinton email dataset revealed by Wikileaks, available at https://www.kaggle.com/kaggle/hillary-clinton-emails
14. Liu, Y., Gummadi, K. P., Krishnamurthy, B., and Mislove, A. Analyzing facebook privacy settings: user expectations vs. reality. In ACM SIGCOMM, 2011.
15. Mazurek, M.L., Liang, Y., Melicher, W., Sleeper, M., Bauer, L., Ganger, G.R., Gupta, N., and Reiter, M.K. Toward strong, usable access control for shared distributed data. In USENIX conference on File and Storage Technologies (FAST), 2014.
16. Mortier, R., Zhao, J., Crowcroft, J., Wang, L., Li, Q., Haddadi, H., Amar, Y., Crabtree, A., Colley J., Lodge, T., Brown, T., McAuley, D., and Greenhalgh, C. Personal Data Management with the Databox: What's Inside the Box? In ACM Workshop on Cloud-Assisted Networking, 2016.
17. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data.
18. Roth, M., Ben-David, A., Deutscher, D., Flysher, G., Horn, I., Leichtberg, A., Leiser, N., Matias, Y., and Merom, R. Suggesting friends using the implicit social graph. In ACM SIGKDD, 2010, 233-242.
19. 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.
20. 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.
21. Wang, F., Mickens, J., Zeldovich, N., and Vaikuntanathan, V. Sieve: Cryptographically Enforced Access Control for User Data in Untrusted Clouds. In USENIX Symposium on Networked Syst. Design and Implem. (NSDI), 2016.
22. Yuan, L., Korshunov, P., and Ebrahimi T. Privacy-preserving photo sharing based on a secure JPEG. In IEEE Conference on Computer Communications Workshops, 2015.
23. Zimmerman, P. PGP user's guide (1994).