SWYSWYK: a new Sharing Paradigm for the Personal Cloud

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

[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, the Personal Cloud emerges and holds 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. This demonstration illustrates a new sharing paradigm, called SWYSWYK (*Share What You See with Who You Know*), dedicated to the Personal Cloud context. It allows each user to physically visualize the net effects of sharing rules and automatically provides tangible guarantees about the enforcement of the defined sharing policies. The usage and internals of SWYSWYK are demonstrated on a running prototype combining a commercial Personal Cloud platform, Cozy, and a secure hardware reference monitor, PlugDB.

Keywords: Personal Cloud, Privacy-by-Design, Access Control, Data Security

1 Introduction

The ever increasing centralization of personal data on servers exacerbates the risk of privacy leakage due to piracy and opaque business practices. Today, a rebalancing of personal data management is occurring worldwide thanks to legislation evolution [1] and smart disclosure initiatives (e.g., blue and green buttons in the US, MesInfos in France). This enables individuals to get their personal data back from companies or administrations and organize it in a Personal Information Management System (PIMS) [2] and share it with applications and users under their control.

But 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 [3] are geared towards central authorities and their secure enforcement requires a deep expertise, out of reach of individuals. Conversely, decentralized tools (e.g., Web of Trust models or FOAF dissemination rules [4, 5, 6]) put on individuals the burden of defining manually each basic sharing rule and leave them on their own to manage complex cryptographic protection against piracy [7]. This often leads to consider data sharing as an intractable burden, letting desperate owners

either define far too permissive policies [8] or 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.

This demonstration proposes an answer to solve this issue. It capitalizes upon 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 [9]. A reference architecture providing tangible guarantees about the enforcement of SWYSWYK policies has been discussed in [10]. We demonstrate how the SWYSWYK paradigm can be put in practice, by combining an open-source PIMS platform, Cozy[1], with a reference monitor embedded in secure hardware, PlugDB[2].

2 SWYSWYK Baseline

SWYSWYK is not yet another access control model. Rather, the access control policy is defined by the PIMS platform. It simply does the assumption that the sharing granularity is the document and that every shareable document is made viewable by the PIMS owner. We do not accept a smaller granularity as it could make the visualization difficult. Every user (called subject hereafter) the owner wants to interact with is also assumed to correspond to a PIMS viewable document (e.g., a contact record). The third and last assumption is that the access control policy is materialized by a set of Access Control List (ACL, or permissions) 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 has the ability to check these ACLs and freely filter out those which presumably hurt her privacy thanks to administration tools detailed next. This principle gives substance to the Share What You See with Who You Know (SWYSWYK) principle. This is in frontal opposition with approaches where sharing policies are defined by a set of potentially complex rules, evaluated on the fly by an opaque reference monitor. The logic of a SWYSWYK reference monitor can be trivially understood by anyone: operation a on d is granted to s iff $(s,d,a) \in ACL$.

The tricky point of the SWYSWYK paradigm lies in the detection and validation of suspicious ACLs. Indeed, permissions are created through a genuine access control model, provided by the PIMS, on which no security assumption can be made.

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 to store them in ACL^+ if she considers them innocuous or in ACL^- otherwise.

[1] https://cozy.io/en/
[2] https://project.inria.fr/plugdb/

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. This mechanism is based on the assumption that owners exhibit a rather stable data disclosure behavior over time [11]. The second mechanism uses *watchdog triggers* highlighting ACLs concerning sensitive documents (e.g., "which new subjects have access to my medical records?"), sensitive subjects (e.g., "which new documents can be seen by my manager?") or sensitive associations (e.g., "which new authorizations my colleagues have on my family photos?"). This mechanism is further detailed in [10].

3 SWYSWYK Architecture

The objective of this architecture is to protect the owner's data by construction against any form of confidentiality attacks. Presented in Fig. 1, it 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. In this demonstration, the *UE* is represented by a personal computer, with a Cozy instance running on it, using an Internet connection. Cozy is a representative open-source PIMS suite, gathering personal data from multiple sources. The *IE* is a Raspberry Pi 3 without any network connection. Finally, the *SEE* is played by PlugDB, a secure and open hardware/software platform developed at Inria. It combines a smartcard to store cryptographic secrets, a microcontroller (MCU) running a relational database engine [12] and a microSD flash card storing the database with crypto-protection against snooping and tampering. It communicates with both the *UE* (WiFi IEEE 802.11n) and the *IE* (High Speed USB 2.0).

PIMS data system. In Cozy, one can install web app created by any developer or third-party on which none security assumption can be made. 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.

Reference monitor. The reference monitor executes the *Allowed* function which grants or denies access. It must be part of the secure core of the architecture and embedded into the *SEE* to guarantee that it cannot be bypassed, observed nor corrupted by any external application. It acts as an incorruptible doorkeeper.

Administration console. The administration console is used to help the PIMS owner to perform the ACL validation task. This console runs the *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 must be isolated in the *IE* to prevent any information leakage.

Internal data structures. The ACL^{*}, $ACL^{?}$, ACL^{+} and ACL^{-} sets, the document metadata and the encryption keys must all be stored inside *SEE* for obvious security reasons, but also for performances issues, as storing them in the *UE* would incur prohibitive decryption and integrity costs. Each ACL set is stored as a bipartite graph to avoid any combinatorial explosion from large sharing rules. This allows the access control to remains *consistent* by construction (the decision is unique), *complete* (the decision always exists) and can be evaluated in *logarithmic time*.

4 Demonstration scenario[3]

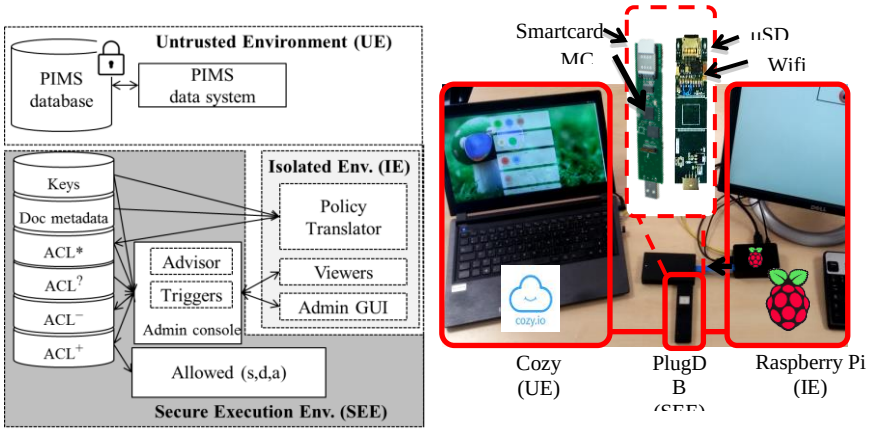

Fig. 1. Demonstration platform.

The first part of the demonstration (see Fig. 2, steps 1-4) shows the utility of SWYSWYK in terms of privacy protection. We use a personal cloud populated by Cozy applications with a set of predefined documents and sharing rules, to which the attendees can connect as Alice (the owner). A remote personal cloud instance, identified as Bob (a subject), is also pre-installed. The attendees first browse Alice's Cozy, and see contact files with pictures representing subjects (among which Bob and Boss, the manager of Alice) and a set of general purpose pictures (among which photos of Alice's 'swollen belly' to be shown to her doctors). The privacy expectations of Alice are that (i) her medical pictures are not shared with anybody and (ii) only innocuous documents are shared with her manager.

The attendees use the 'Files' application on Alice's Cozy and create a new sharing rule through the interface. They are then invited to go in the Administration GUI, to look at the existing sharing rules, represented as logic-based predicates on documents' metadata. Unsurprisingly, this task requires a deep expertise, out of reach of the regular user as the number of rules increases. Alternatively, the GUI shows the corre-

[3] A video of the demonstration is available at : http://wanda.inria.fr/CIKM/cikm.ogg

sponding set of visualizable ACLs, where attendees can discover that Bob has a read access to compromising pictures of Alice - confirmed when connecting to Bob's personal cloud -. However, the size of the ACL set proscribes any exhaustive 'manual' check. The attendees then activate the *Advisor* module to identify suspicious ACLs. They also create (or choose predefined) *watchdog triggers* to automatically detect appearance of sensitive objects (e.g., medical pictures) or subjects (e.g., Boss) in the produced ACLs. As a result, the attendees detect further ACLs hurting Alice's privacy, which can be rejected with direct effect on Bob's Cozy.

The second part of the demonstration (Fig. 3, steps 1-4 and step 5) focuses on the security properties and shows what is happening 'under the hood' at rule creation and execution time. The attendees play the attacker role and add a malicious rule hurting Alice privacy. The GUI shows what is running in the three environments, where and when data is decrypted, keys are stored and access decisions are taken.

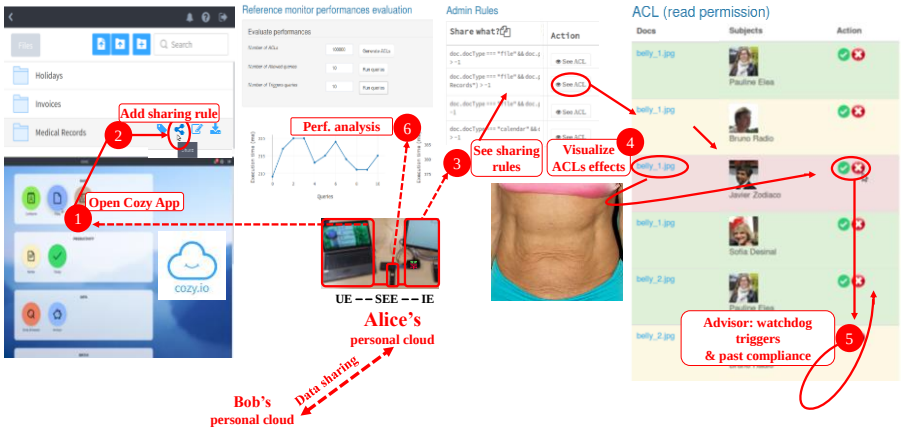

Fig. 2. Demonstration scenario and GUI.

The third part of the demonstration (Fig. 3, step 6) focuses on performance, scalability and compliance with constrained secure execution environments like PlugDB. We use an ACL generator, producing large sets of ACLs combining existing subjects and objects. The performances of the reference monitor are plotted along with detailed statistics about execution times, cryptographic impact and I/O costs.

5 Conclusion

This demonstration illustrates the three following properties attached to the SWYSWYK paradigm.

Usability. The focus of the demonstration is on validating and sanitizing the resulting set of ACLs. It highlights the effectiveness of *watchdog triggers* and the *Advisor* process to help the owner filtering out compromising ACLs, even when applied to a large set of candidate ACLs.

Security by construction. The combination of document encryption in *UE*, code isolation in *IE* and a tamper-resistant *SEE* provides built-in guarantees to the owner against confidentiality attacks, out of reach of traditional approaches. Moreover, validating the approach in a highly constrained tamper-resistant environment like PlugDB is a proof of the simplicity of the reference monitor and of the associated administration tools. This simplicity also opens the way to a formal proof of the embedded code.

Performance. The Cozy queries executions are slowed down due to encryption and data communication. However, the overhead is kept reasonable for a regular use. This validates the practicality of the approach while large performance gains can be expected with environments providing higher communication throughput.

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 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.
2. Abiteboul, S., André, B., Kaplan, D. Managing your digital life. In: Communications of the ACM (CACM), 58(5), 32-35 (2015).
3. Bertino, E., Ghinita, G., and Kamra, A. Access control for databases: Concepts and systems. In: Foundations and Trends in Databases (2011).
4. Bellavista, P., Giannelli, C., et al. Peer-to-Peer Content Sharing Based on Social Identities and Relationships. In: IEEE Internet Computing, 18(3), 55-63 (2013).
5. Carminati, B., Ferrari, E., and Perego, A. Rule-Based Access Control for Social Networks. In: On the Move to Meaningful Internet Systems, pp. 411-415 (2006).
6. Van Kleek, M., Smith, D.A., Shadbolt, N. et al. A decentralized architecture for consolidating personal information ecosystems: The WebBox. In: PIM (2012).
7. 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 & Implem. (2016)
8. Liu, Y., Gummadi, K. P., Krishnamurthy, B., Mislove, A. Analyzing facebook privacy settings: user expectations vs. reality. In: Proceedings of ACM SIGCOMM conference on Internet measurement conference, pp. 61-70 (2011).
9. Tran-Van, P., Anciaux, N., Pucheral, P. A new Sharing Paradigm for the Personal Cloud. In: Trust, Privacy & Security in Digital Business, pp. 180-196 (2017).
10. Tran-Van, P., Anciaux, N., Pucheral, P. SWYSWYK: a Privacy-by-Design Paradigm for Personal Information Management Systems. To appear at the 26th International Conference on Information Systems Development (2017).
11. Roth, M., Ben-David, A., Deutscher, D., Flysher, G., Horn, I., et al. Suggesting friends using the implicit social graph. In: Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 233-242 (2010).
12. Anciaux, N., Bouganim, L., Pucheral, P., Guo, Y., Le Folgoc, L., and Yin, S. MILo-DB: a personal, secure and portable database machine. In: Distributed and Parallel Databases, 32(1), 37-63 (2014).