HAL open science

PriMan : A Privacy-Preserving Identity Framework

Kristof Verslype, Pieter Verhaeghe, Jorn Lapon, Vincent Naessens, Bart Decker

To cite this version:

Kristof Verslype, Pieter Verhaeghe, Jorn Lapon, Vincent Naessens, Bart Decker. PriMan : A Privacy-Preserving Identity Framework. 24th Annual IFIP WG 11.3 Working Conference on Data and Applications Security and Privacy (DBSEC), Jun 2010, Rome, Italy. pp.327-334, 10.1007/978-3-642-13739-6_24 . hal-01056676

HAL Id: hal-01056676

https://inria.hal.science/hal-01056676v1

Submitted on 20 Aug 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

PriMan: A Privacy-Preserving Identity Framework

Kristof Verslype[1], Pieter Verhaeghe[1], Jorn Lapon[2], Vincent Naessens[2], and Bart De Decker[1]

[1] Katholieke Universiteit Leuven, Department of Computer Science,
Celestijnenlaan 200A, 3001 Heverlee, Belgium,
firstname.lastname@cs.kuleuven.be
[2] Katholieke Hogeschool Sint-Lieven, Department of Industrial Engineering
Gebroeders Desmetstraat 1, 9000 Gent, Belgium,
firstname.lastname@kahosl.be

Abstract. *PriMan* is presented; privacy-preserving user-centric identity management middleware which defines and groups the required functionality. It offers the application developer a uniform technology-agnostic interface to use and combine different types of privacy enhancing technologies. Moreover, the *PriMan* framework defines all the components and their functionality required to raise the development of privacy enhanced client-server applications to a higher level.

1 Introduction

The digitalization of our society comes with a lot of benefits. However, privacy of the user is increasingly at stake. The awareness of both citizens and companies is rising. In fact, both can benefit from a higher level of privacy in applications.

Therefore, techniques are being developed to improve the user's anonymity; crowds and mix networks at network level, and anonymous credentials w.r.t. personal user properties. The latter enable to prove only the required properties, e.g. that you are older than 18 if one of the credential attributes is your date of birth.

The privacy enhancing technologies (PETs) are heterogeneous in approach; e.g., pseudonym certificates are sent to the verifier, while Idemix credentials only send proofs. Hence, it will cost the application developer much effort to develop a privacy-friendly application, especially when multiple credential types must be supported. Also, chances are that the privacy issues will be omitted or that the privacy is inadequately protected due to incorrect use of PETs. Moreover, even in privacy-friendly applications, the user remains in the dark about to whom and under which pseudonym personal properties have been disclosed or, in short, about his degree of anonymity towards others.

Therefore, *PriMan*, a privacy-preserving user-centric identity middleware framework is designed and implemented. It facilitates the development of privacy-enhanced applications. The different credential approaches are reconciled, resulting in a uniform interface enabling the application developer to choose the most appropriate technology and to easily switch to another one when e.g. the requirements change.

Three privacy-preserving applications in three different domains have been built on top of this framework; an ePoll (eGoverment), an eTicketing (eCommerce) and an

ePrescription (eHealth) application. The design of the presented framework has been reiterated several times driven by the feedback received from the application developers.

The next section briefly sums up the relevant privacy preserving technologies supported by the framework. The requirements derived from the applications are presented in section 3. The general architecture, the generic credential representation and the validation are given in section 4, 5 and 6. Related work is discussed in section section 7 and the conclusions are given in section 8.

2 Building Blocks

The supported building blocks and the main credential systems are touched.

Mix networks (e.g. [1]) and **crowds** (e.g. [2]) guarantee anonymity at network level. A **commitment** scheme [3, 4] allows an entity to commit to a set of values, while keeping these secret. The commitment hides the values towards the verifier, but allows the creator to prove properties of the committed values. A **verifiable encryption** scheme (e.g., [5]) also allows the creator to prove properties about the encrypted values, while the verifier is ensured that a known TTP will be able to decrypt the ciphertext if necessary. A **Pseudonym** is an identifier presumably unlinkable to a real identity.

An **X.509 certificate** is a set of personal attributes and other related properties signed by a certifier using a standard signing algorithm. The certificate owner needs the corresponding private key to prove ownership of it. Presenting it to others implies disclosing all the content in the certificate. X.509 certificates are revoked by adding their serial numbers on a revocation list. **Pseudonym certificates** [6] are standard certificates in which the identity information is replaced with a pseudonym. Different shows of the same certificate are linkable, potentialy undermining anonymity. The privacy can further be increased by substituting hashes or MACs for the actual attribute values. This way the certificate owner can decide which attributes to disclose. An **anonymous credential** [7, 8] allows for selective disclosure of properties of credential attributes, while hiding the others. The credential itself with its values is not revealed, but instead, a zero-knowledge proof of knowledge that the disclosed properties were certified by the issuer. Usages of anonymous credentials can either be linkable (e.g. UProve) or unlinkable (e.g. Idemix). It is possible to prove membership of a set without revealing anything else. The disclosed properties can also involve attributes in other credentials, in verifiable encryptions and in commitments. Idemix credentials can be shown under a pseudonym to which the credential is not bound. The issuer of an Idemix credential can set a global limit on the number of times the credential can be used. Finally, a credential usage can optionally be deanonymized afterwards by a trusted party in case of abuse.

3 Requirements

The framework specific requirements are formulated (Fx) and are followed by the framework tasks derived from the applications (Tx). T1-T3 are indispensable. T4-T6 are needed to build a full-fledged privacy-preserving identity framework.

F1. User-centricity. The user controls the disclosure of his personal data.

F2. Usability. A technology agnostic, intuitive interface facilitates the development of privacy preserving applications and plugging in new implementations (e.g. UProve credentials) must be easy.

F3. Modularity. By loading only the required modules and implementations, *PriMan* can be run on portable devices; e.g. on a doctor's portable device to issue prescriptions on location in the ePrescription application.

F4. Protection of (highly) confidential information. Confidential data can be secret keys, but also personal data, since it can reduce users' anonymity.

T1. Setting up connections with various properties. An SSL connection might suffice for e.g. registration. However, a mix network might be a better choice to protect the user's anonymity for e.g. anonymous poll signing.

T2. Creation and usage of credentials. In all three applications, credentials of different types are issued and used. Proving properties during a credential show often requires commitments and verifiable encryptions.

T3. Secure storage of credentials & credential related data. Users often have many credentials, which must be stored and managed securely. Some credentials should always and everywhere be available and, hence, must be stored on a smartcard or on a remote server (e.g. ticketing).

[T4.] Anonymity set estimation. The user discloses mandatory properties in the ePoll and eTicketing plus potentially optional ones. The framework must estimate the consequences of these disclosures on the user's anonymity and give advice.

[T5.] Profile tracking. In the eTicketing application, multiple purchases by a user of tickets for the same event need to be linkable in order to be able to restrict the maximum number of tickets per customer. If the user discloses different properties when buying tickets at different occasions, his anonymity may decrease. Therefore, the framework has to securely and locally keep track of the properties disclosed under pseudonyms to other parties.

[T6.] Dispute solving. In the ePrescription and eTicketing application, abuse is possible. Hence, support for deanonymization must be provided.

4 General Architecture

The above-formulated requirements led to the *PriMan* architecture of which a high level overview is presented in figure 1. *PriMan* consists of abstract handler interfaces and concrete managers. A handler interface provides a uniform interface to a class of technologies such as credentials, connections or storage. A handler is in general a wrapper around an existing implementation of a technology (e.g. Idemix). A provider contains concrete handlers. Multiple providers can be plugged into *PriMan*. Each of the first six managers corresponds to one of the framework tasks T1-T6 and keeps track of and uses the underlying concrete handlers to offer higher level functionality, since the existing technologies are rather low level. A special manager is the policy manager to automate decisions. The `PriManFacade` is the application's entry point to the managers. Also, an appropriate GUI can be implemented and loaded; e.g. one GUI for PDAs and one for desktop computers.

A **connection hander** sets up, listens for and closes connections (T1). The **connection manager** allows the developer to specify connection properties such as integrity

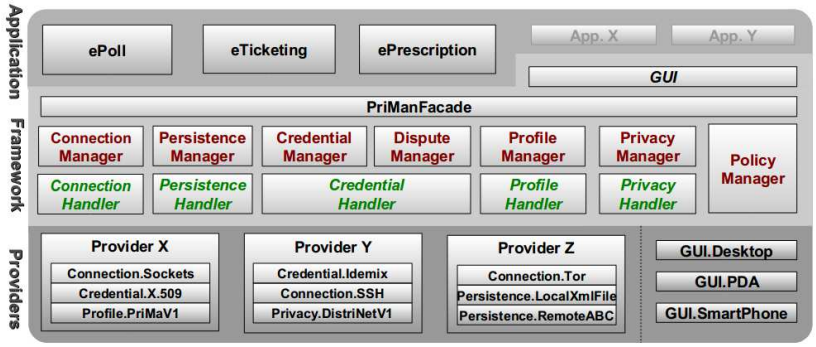

Fig. 1. High level architecture of *PriMan*.

and/or anonymity properties. Based on these properties an appropriate handler is selected to set up the connection.

A **credential handler** provides the functionality to issue and receive credentials and pseudonyms as well as to authenticate or to sign messages with these credentials (T2). Three subhandler interfaces deal with revocation, commitments and verifiable encryptions. The service provider's access policy will define which credentials are to be used and which properties the user must or may disclose. Based on this request, the user's **credential manager** obtains the sets of credentials, commitments, pseudonyms and verifiable encryptions able to fulfill the request.

Storage of credentials and credential related data such as commitments (T3) is done by the **persistence manager**. It keeps track of where the different data objects are stored and which handler maintains them. Each **handler** defines a location type (e.g. smart card or server), an encoding structure (e.g. XML) and the protection mechanism (e.g. password based).

The **privacy manager** estimates the anonymity of pseudonyms towards other parties and the impact of disclosing certain properties (T4). Each **privacy handler** provides a concrete metric therefore.

Profile tracking (T5) is done by the **profile manager**. The **profile handler** keeps track of one or more profiles. Depending on the framework policy, authorization can be given to external entities to do certain types of queries on one or more profiles: adding or requesting for data which can be application or context specific (e.g. books bought). Also, a user can add data to a profile (e.g. books (s)he is interested in). The **profile manager** determines to which profile data are added. A **profile handler** also implements heuristics to probabilistically link profiles.

The **Dispute Manager** (T6) offers the means to file complaints in case of abuse. Complementary, evidence to protect against false accusations can be stored and later be disclosed to trusted third parties. These parties can do the deanonymization when certain conditions are fulfilled. Since deanonymization is credential type specific, an underlying credential handler is used.

Each provider consists of a set of concrete handlers (e.g. Credential.X509 and Connection.Tor). For each implementation, the provider maintains some bookkeeping information (names, properties, versions, ...)

5 Generic Credential Representation

The central concept in the framework are credentials. Therefore, this section presents a uniform, technology agnostic representation of credentials and related objects, which facilitates switching to other technologies. Credential technologies with heavily differing approaches such as passwords, X.509 certificates and Idemix credentials fit in the representation. Different object types are defined.

Credential template. It describes everything credentials of a certain category have in common (e.g. Belgian driving licenses with the same issuer public key): 1) technology specific *security parameters* such as key lengths; 2) *control settings* defining the credential's validity and usage rights such allowance to sign, the show limit, validity period and credential verification info; 3) the *issuer* data and 4) an *attribute specification* which specifies mainly the label and type of each of the attributes.

Credential. A credential consists of 1) a *credential template*, 2) *credential values*; i.e. credential's attribute values and the validity start date, 3) a *credential trace* containing all the information disclosed each time the credential is used (e.g. serial number and public key), and 4) (references to) *credential secrets* required to use the credential. Credentials never leave the framework. Credential secrets and values are sensitive data and the latter can only be exported by a framework protocol.

Show specification. This describes the properties to disclose or that were disclosed when a credential is/was used to sign, verify or issue a credential.

Disclosure. This contains the show specification and the involved objects required to either prove or verify the properties described in the show specification. Multiple credentials, commitments and verifiable encryptions and a pseudonym can be involved. In addition, deanonymization specifications can be added to specify the deanonymizers and deanonymization conditions (typically abuse). When a prover sends a disclosure to the verifier, information such as secrets and attribute values are removed from the contained objects, such that the received disclosure can only be used for verifications.

Entity. An entity represents a person or organisation and consists of a verifiable disclosure together with a proof (authentication or signature) of this disclosed information. Entities are useful to keep track of the information known to or about others. It allows to verify certification chains, which consist of entities.

Transcript. This is a framework protocol return value which contains all exchanged data and data required to rerun the protocol such as connection id or an entity representing the prover in case of an authentication verification. Transcripts are profile manager input.

The control settings in the templates allow for multiple issue keypairs in one credential and, hence, allow for issuing new credentials of different types with one existing credential; for instance, issuance of Idemix and UProve credentials with an X.509 credential. Similarly, multiple verifiable encryption keypairs can be included.

If the properties in the show specification are proven, the verifier knows in addition the credential trace. Hence, the less information in the credential trace, the better (which is technology dependent). This can be checked by the prover in advance.

As an example, an X.509 certificate without private key is represented as an entity since it represents a person or organization about which properties are certified. The corresponding verifier *disclosure* contains a credential trace and a credential template which contain all the information such as the attribute names and values and the certification signature to let an X.509 handler recompose the original X.509 certificate. An Idemix credential will have an empty credential trace.

6 Validation

In the ePoll application, citizens can participate in multiple polls using an anonymous credential, but can only vote once for each poll. Votes of the same user are unlinkable. The poll organizer can restrict the voter set and can invite voters to disclose some additional properties, enabling the generation of more significant poll statistics. For instance; the poll could be restricted to adults and optionally, they may disclose their gender. Poll signatures are published to allow everyone to verify the poll's correctness. In figure 2, a client and server application use different managers to build this application.

First, the client application creates an anonymous connection with the ePoll server (C1-C3,S1), which replies by sending a request (C4,S2). A `Request` object contains a description of the obligatory and optional properties to be disclosed and can contain a list of sign options. It also lists the templates of acceptable credentials. The received request thus contains the user's different personal property disclosure options and the different choices he can vote for. Based on the request, the client's credential manager is asked to give a description of each credential able to fulfill the request (C5); in the ePoll case, there is one such credential. Based on the request and the credential description, all the possible show specifications are returned (C6); i.e. all the sets of properties the user can disclose using that credential in order to cast a vote. The application could optionally ask the profile manager for the profile containing the information disclosed previously to that poll service (C7). Since the user never signed the poll, this profile will be empty. The privacy manager is asked for the impact on the privacy for each of the possible show specifications (C8). The returned `PrivacyInfo` object contains the relevant anonymity information for each of the show specifications if the contained properties were disclosed. Now, the application can decide which properties to disclose. Note that the policy manager can already filter out some possibilities.

The user selects the properties to disclose and the option to vote for (C9, C10). The client's credential manager is asked to create the `Disclosure` object corresponding to the show specification (C11). Therefore, it loads the corresponding credential (using the persistence manager) and creates the required pseudonym. The nym, credential and show specification are put in the disclosure, which is used to create the signature (C12). The disclosure, signature and choice are sent to the poll organizer (C13-15,S3-S5). The client's profile manager is informed that over `conn` a `disclosure` has happened, which was in fact a signature on `choice` and that the signature is published (C16).

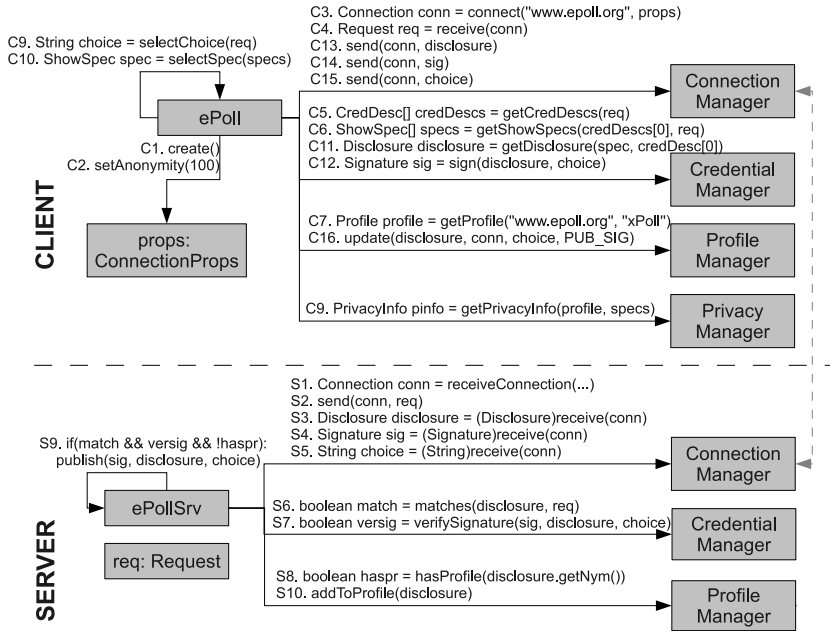

Fig. 2. Example: signing an anonymous poll.

After receiving a signature, disclosure and the corresponding choice (S3-S5), the server application asks the credential manager if the disclosure matches the request (S6) and if the signature is valid (S7) and it asks the profile manager if the pseudonym in the disclosure has not yet been used (S8). If these three conditions are met, the server application publishes the signature, disclosure and signed choice (9).

Note that both client and server use the framework in a complementary way and also notice how easy it is to build applications on top of *PriMan*.

7 Related Work

Several federated identity management systems (FIMs) exist such as Shibboleth, Windows Cardspace, OpenId, Athens and Higgins. They all have in common that the user data is stored by a trusted identity provider (IdP). Some of the federated identity management systems allow the user to request the IdP a token containing only properties of the user, hence, improving the user's privacy. Still the IdP knows when what properties were requested by the user. None of the current FIMs offers real user-centric identity management, which is made possible using anonymous credentials. Also, none of the FIMs offers the user the possibility to keep track of disclosed properties, nor can they inform the user about his/her anonymity status. Since the IdP can always link an issued token to an identity, the FIMs do not need the possibility of deanonymization. Their tokens are typically based on SAML. Our approach is built upon the idea of real

user-centric identity management, centered around the concept of anonymous credentials and offers support for the above-mentioned functionality. In addition, it can offer support for any credential type and is not limited to SAML tokens.

PriMan allows to use different concepts in a coherent way; credentials, connections, persistence, profiles, etc. Hence, the framework integrates current and future implementations and research in one aggregate. For instance, a high-level approach to control the information disclosure based on the sensitivity and the possibility that a user's identity is revealed was proposed [9] and could be implemented by a privacy handler.

8 Conclusions

This paper presented *PriMan*, a flexible middleware framework that considerably facilitates the development of privacy-preserving applications. Although not all building blocks have an implementation yet, *PriMan* is already a very useful tool for developers.

Acknowledgements. This research is partially funded by the Interuniversity Attraction Poles Programme Belgian State, Belgian Science Policy and the Research Fund K.U.Leuven and the IWT-SBO project (ADAPID) "Advanced Applications for Electronic Identity Cards in Flanders".

References

1. Camenisch, J., Mityagin, A.: Mix-network with stronger security. In: Privacy Enhancing Technologies. (2005) 128–146
2. Danezis, G., Díaz, C., Käsper, E., Troncoso, C.: The wisdom of crowds: Attacks and optimal constructions. In: ESORICS. (2009) 406–423
3. Pedersen, T.: Non-interactive and information-theoretic secure verifiable secret sharing. In: CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology, London, UK, Springer-Verlag (1992) 129–140
4. Damgard, I., Pedersen, T., Pfitzmann, B.: Statistical secrecy and multi-bit commitments (1996)
5. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: CRYPTO '98: Proceedings of the 18th Annual International Cryptology Conference on Advances in Cryptology, London, UK, Springer-Verlag (1998) 13–25
6. Asokan, N., Herreweghen, E.V., Steiner, M.: Towards a framework for handling disputes in payment systems. Technical Report RZ 2996 (1998)
7. Camenisch, J., Lysyanskaya, A.: An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In: EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques, London, UK, Springer-Verlag (2001) 93–118
8. Brands, S.: Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy. MIT Press, Cambridge, MA, USA (2000)
9. Irwin, K., Yu, T.: An identifiability-based access control model for privacy protection in open systems. In: WPES '04: Proceedings of the 2004 ACM workshop on Privacy in the electronic society, New York, NY, USA, ACM (2004) 43–43