Privacy-preserving Digital Identity Management for Cloud Computing

Elisa Bertino
CS Department
Purdue University
West Lafayette, Indiana
bertino@cs.purdue.edu

Federica Paci
CS Department
Purdue University
West Lafayette, Indiana
paci@cs.purdue.edu

Rodolfo Ferrini
CS Department
Purdue University
West Lafayette, Indiana
rferrini@purdue.edu

Ning Shang
CS Department
Purdue University
West Lafayette, Indiana
nshang@cs.purdue.edu

Abstract

Digital identity management services are crucial in cloud computing infrastructures to authenticate users and to support flexible access control to services, based on user identity properties (also called attributes) and past interaction histories. Such services should preserve the privacy of users, while at the same time enhancing interoperability across multiple domains and simplifying management of identity verification. In this paper we propose an approach addressing such requirements, based on the use of high-level identity verification policies expressed in terms of identity attributes, zero-knolwedge proof protocols, and semantic matching techniques. The paper describes the basic techniques we adopt and the architeture of a system developed based on these techniques, and reports performance experimental results.

1 Introduction

Internet is not any longer only a communication medium but, because of the reliable, afforbable, and ubiquitous broadband access, is becoming a powerful computing platform. Rather than running software and managing data on a desktop computer or server, users are able to execute applications and access data on demand from the "cloud" (the Internet) anywhere in the world. This new computing paradigm is referred to as *cloud computing*. Examples of cloud computing applications are Amazon's Simple Storage Service (S3), Elastic Computing Cloud (EC2) for storing photos on Smugmug an on line photo service, and Google Apps for word-processing.

Cloud services make easier for users to access their personal information from databases and make it available to services distributed across Internet. The availability of such information in the cloud is crucial to provide

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

better services to users and to authenticate users in case of services sensitive with respect to privacy and security. Users have typically to establish their identity each time they use a new cloud service, usually by filling out an online form and providing sensitive personal information (e.g., name, home address, credit card number, phone number, etc.). This leaves a trail of personal information that, if not properly protected, may be misused. Therefore, the development of digital identity management (IdM for short) systems suitable for cloud computing is crucial. An important requirement is that users of cloud services must have control on which personal information is disclosed and how this information is used in order to minimize the risk of identity theft and fraud.

Another major issue concerning IdM in cloud platforms is interoperability. Interoperability issues range from the use of different identity tokens, such those encoded in X.509 certificates and SAML assertions, and different identity negotiation protocols, such as the client-centric protocols and the identity-provider centric protocols, to the use of different names for identity attributes. An *identity attribute* encodes a specific identity information about an individual, such as the social-security-number; it consists of an attribute name, also called identity tag, and a value. The use of different names for identity attributes, that we refer to as *naming heterogeneity*, typically occurs whenever users and cloud service providers use different vocabularies for identity attribute names. In this case, whenever a cloud service provider requests from a user a set of identity attributes to verify the user identity, the user may not understand which identity attributes he/she has to provide.

To address the problem of privacy-preserving management of digital identity attributes in domains with heterogeneous name spaces, we propose a privacy-preserving multi-factor identity attribute verification protocol supporting a matching technique based on look-up tables, dictionaries, and ontology mapping techniques to match cloud service providers and clients vocabularies. The protocol uses an aggregate zero knowledge proofs of knowledge (AgZKPK) cryptographic protocol to allow clients to prove with a single interactive proof the knowledge of multiple identity attributes without the need to provide them in clear.

The rest of the paper is organized as follows. Section 2 introduces the notions on which our multi-factor identity attribute verification protocol is based. Section 3 presents the multi-factor identity attribute verification protocol. Section 4 presents the system architecture and discuss implementation while Section 5 reports experimental results. Finally, Section 6 concludes the paper and outlines some future work.

2 Preliminary concepts

Our approach, as many other approaches, assumes an IdM system that include several entities: Identity Providers (IdPs), Cloud Service Providers (CSPs), Registars, and users. CSPs provide access to data and software that reside on the Internet. IdPs issue certified identity attributes to users and control the sharing of such information. Registrars are additional components that store and manage information related to identity attributes used in our multi-factor identity attribute verification approach. Note that, unlike the IdPs, the information stored at the Registrars does not include the values of the identity attributes in clear. Instead, such information only contains the cryptographic semantically secure commitments [1]of the identity attributes which are then used by the clients to construct zero knowledge proofs of knowledge (ZKPK) [2]of those attributes. The Registrar stores for each user an Identity Record (IdR) containing an identity tuple for each user's identity attribute m. Each identity tuple consists of a tag, that is, an attribute name, the Pedersen commitment [5] of m, denoted by M_i, the signature of the registrar on M, denoted by σ_i, two types of assurance, namely *validity assurance* and *ownership assurance*, and a set of nyms (also called weak identifiers) $\{W_{ij}\}$[3]. M_i is computed as $g^m h^r$, where m is the value of the

[1]A commitment scheme or a bit commitment scheme is a method that allows a user to commit to a value while keeping it hidden and preserving the user's ability to reveal the committed value later.

[2]A zero-knowledge proof or zero-knowledge protocol is an interactive method for one party to prove to another that a (usually mathematical) statement is true, without revealing anything other than the veracity of the statement.

[3]Nyms are used to link together different identity tuples of the same individual for multi-factor authentication. Nyms do not need to be protected.

identity attribute, r is a random number in $\mathcal{Z}_p$ and only known to the client, and g and h are generators of a group G of prime order p. G, g, h and p are public parameters of the Registrar. Validity assurance corresponds to the confidence about the validity of the identity attribute based on the verification performed at the identity attribute original issuer. Ownership assurance corresponds to the confidence about the claim that the principal presenting an identity attribute is its true owner. The identity tuples of each registered client can be retrieved from the Registrar by CSPs (*online* mode) or the Registrar can release to the client a certificate containing its identity record (*offline* mode).

3 Interoperable Multi-Factor Authentication

Our multi-factor authentication protocol takes place between a client and a CSP and consists of two phases. In the first phase, the CSP matches the identity attributes in the clients vocabulary with its own attributes to help the client understand its identity verification policy. An *identity verification policy* consists of the set of identity attributes that the user must prove to know; if the values of these identity attributes are only needed for verification purposes but not for the execution of the service required by the client, the CSP has no reason to have to see these values in clear. In the second phase, the client executes the AgZKPK protocol to prove the CSP the knowledge of the matched identity attributes. The use of this protocol allows the client to convince the CSP that the client knows the values of the identity attributes without having to reveal to the CSP the values in clear.

3.1 The protocol for identity attribute matching

Our attribute name matching technique uses a combination of look-up tables, dictionaries, and ontology mapping in order to address the different variations in identity attribute names. *Syntactic variations* refer to the use of different character combinations to denote the same term. An example is the use of "CreditCard" and "Credit_Card". *Terminological variations* refer to the use of different terms to denote the same concept. An example of terminological variation is the use of the synonyms "Credit Card" and "Charge Card". *Semantic variations* are related to the use of two different concepts in different knowledge domains to denote the same term. Syntactic variations can be identified by using look up tables. A look up table enumerates the possible ways in which the same term can be written by using different character combinations. Terminological variations can be determined by the use of dictionaries or thesaurus such as WordNet [6]. Finally, semantic variations can be solved by ontology matching techniques. An ontology is a formal representation of a domain in terms of concepts and properties relating those concepts. Ontologies can be used to specify a domain of interest and reason about its concepts and properties. Ontology mapping is the process whereby the concepts of an ontology - the source ontology - are mapped onto the concepts of another ontology - the target ontology - according to those semantic relations [4].

An important issue related to the identity matching protocol is which party has to execute the matching. In our approach the matching is performed by the CSP, in that performing the matching at the client has the obvious drawback that the client may lie and asserts that an identity attribute referred to in the CSP policy matches one of its attribute, whereas this is not the case. The use of ZKPK protocols (see next section) preserves the privacy of the user identity attributes by assuring that the CSP do not learn the values of these attributes; thus the CSP has no incentive to lie about the mapping. A second issue is how to take advantage of previous interactions that the client has performed with other CSPs. Addressing such issue is crucial in order to make the interactions between clients and CSPs fast and convenient for the users. To address such issue, the matching protocol relies on the use of *proof-of-identity certificates*; these certificates encode the mapping between (some of) the user identity attributes and the identity attributes referred in the policies of CSPs with which the user has successfully carried out past interactions.

Let *AttrProof* be the set of identity attributes that a CSP asks to a client in order to verify the identity of the

user on behalf of which the client is running. Suppose that some attributes in *AttrProof* do not match any of the attributes in $AttrSet$, the set of clients' identity attributes. We refer to the set of component service's identity attributes that do not match a client attribute name to as *NoMatchingAttr*. The matching process consists of two main phases. The first phase matches the identity attributes that have syntactical and terminological variations. During this phase, the CSP sends to the client, for each identity attribute a_i in the *NoMatchingAttr* set, the set *Synset*$_i$ containing a set of alternative character combinations and a set of synonyms. The client verifies that for each identity attribute a_i, there is an intersection between *Synset*$_i$ and *AttrSet*. If this is the case attribute a_i is removed from *NoMatchingAttr*. Otherwise, if *NoMatchingAttr* is not empty, the second phase is performed. During the second phase the client sends *CertSet*, the set of its proof-of-identity certificates to the CSP. Thus, in the second phase of the matching process the CSP tries to match the concepts corresponding to the identity attributes the client is not able to provide with concepts from the ontologies of the CSPs which have issued the proof-of-identity certificates to the client. Only matches that have a confidence score s greater than a predefined threshold, set by the CSP, are selected. The greater the threshold, the greater is the similarity between the two concepts and thus higher is the probability that the match is correct. If the CSP is able to find mappings for its concepts, it then verifies by using the information in the proof-of-identity certificates that each matching concept matches a client's attribute *Attr*. If this check fails, the CSP terminates the interaction with the client.

3.2 Multi-factor authentication

Once the client receives *Match*, the set of matched identity attributes from the CSP, it retrieves from the Registrar or from its *RegCert*, that is a certificate repository local to the client, the commitments M_i satisfying the matches and the corresponding signatures σ_i. The client aggregates the commitments by computing $M = \prod_{i=1}^{n} M_i = g^{m_1+m_2+...+m_i} h^{r_1+r_2+...+r_i}$ and the signatures into $\sigma = \prod_{i=1}^{n} \sigma_i$, where σ_i is the Registrar 's signature on the committed value $M_i = g^{m_i} h^{r_i}$. According to the ZPK protocol, the client randomly picks y, s in $[1, ..q]$, computes $d = g^y h^s \pmod p$, and sends d, σ, M, M_i , $1 \leq i \leq t$, to the CSP. The CSP sends back a random challenge $e \in [1, .., q]$ to the client. Then the client computes $u = y + em \pmod q$ and $v = s + er \pmod q$ where $m = m_1 + \ \ldots \ m_t$ and $r = r_1 + \ \ldots \ r_t$ and sends u and v to the CSP. The CSP accepts the aggregated zero knowledge proof if $g^u h^v = dc^e$. If this is the case, the CSP then checks that $\sigma = \prod_{i=1}^{n} \sigma_i$. If also the aggregate signature verification succeeds, the CSP releases a proof-of-identity certificate to the client. The certificate states that client's identity attributes in the *Match* set are mapped onto concepts of the CSP ontology and that the client has successfully proved the knowledge of those attributes. The CSP sends the proof-of-identity certificate to the client and stores a copy of the certificate in its local repository *CertRep*. The proof-of-identity certificate can be can be provided to another CSP to allow the client to prove the knowledge of an attribute without performing the aggregate ZKP protocol. The CSP that receives the certificate has just to verify the validity of the certificate.

Example 1: Assume that a user Alice submits a request to her *Hospital Web portal* to access her test results. The *Hospital Web portal* retrieves the test results through the *Laboratory service*. The *Laboratory service* has to verify the identity of Alice in order to provide her test results. The identity verification policy of the *Laboratory service* requires Alice to provide *Medical Assurance*, *Social Security Number* and *Patient ID* identity attributes. Alice provides the aggregated proof of the required identity attributes to the *Hospital Web portal* which forwards them to the *Laboratory service*. The *Laboratory service* then verifies by carrying out an aggregate ZPK protocol with Alice that she owns the required attributes and releases a proof-of-identity certificate. Such certificate asserts Alice is the owner of the *Medical Assurance*, *Social Security Number* and *PatientID* identity attributes she has presented. The next time Alice would like to access her test results through *Hospital Web portal* portal she will present the proof-of-identity certificate to the *Hospital Web portal* which will forward the certificate to the *Laboratory service*. The *Laboratory service* will verify the validity of Alice's certificate and return the test results to the *Hospital Web portal* which will display the results to Alice.

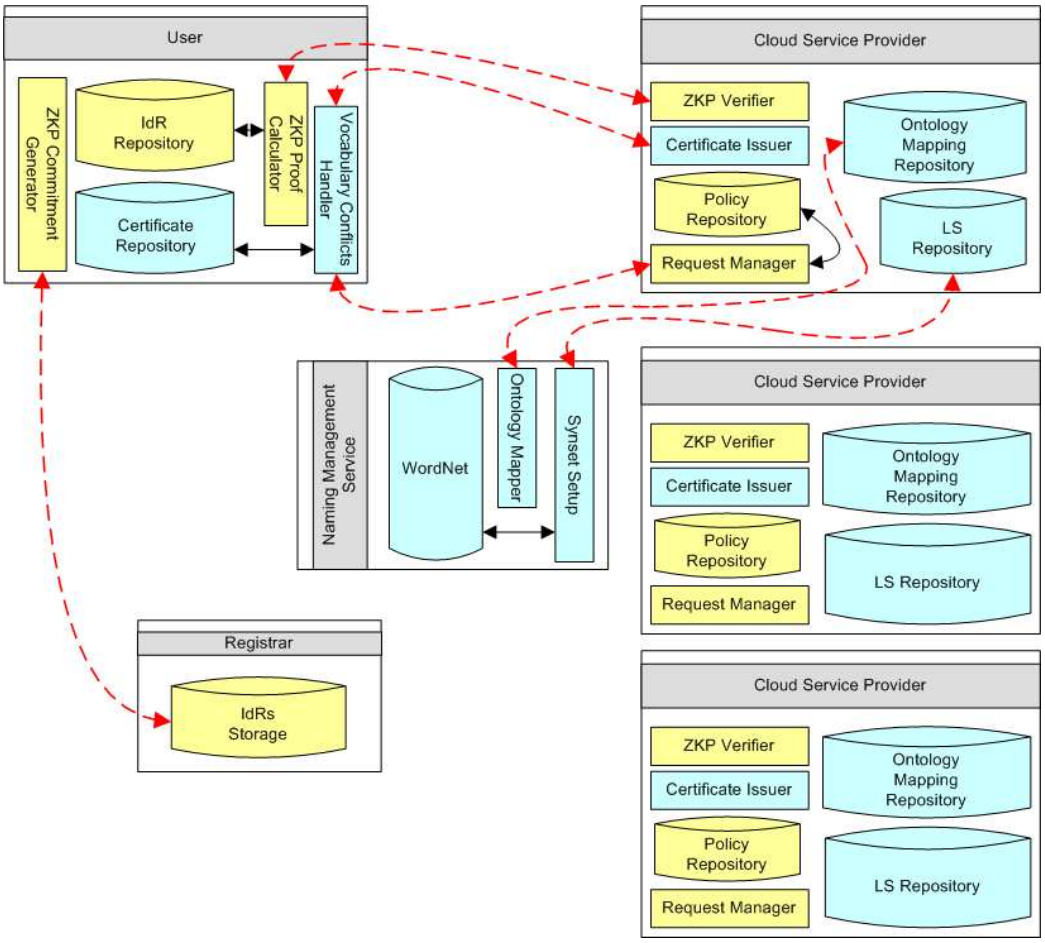

Figure 1: System architecture

4 System architecture and Implementation

In this section we discuss the system architecture that supports our multi-factor identity attributes authentication for cloud services. The architecture consists of four main components: the Registrar, the Service Provider, the User, and the Heterogeneity Management Service. The Registrar component manages the client's identity records and provide functions for retrieving the public parameters required by the AgZKPK protocol. The User component consists of three main modules: the ZKP Commitment Generator, the ZKP Proof Calculator, and the Vocabulary Conflicts Handler. The ZKP Commitment Generator provides the functions for computing the Pedersen commitments of identity attributes; the ZKP Proof Calculator generates the AgZKPK to be provided to CSPs, the Vocabulary Conflicts Handler module checks if there are client identity attributes names that matches the Synsets sent by the Service Provider component and manages the proof-of-identity certificates stored in a local repository. The Service Provider is composed of four modules the Request Manager, the Mapping Path Manager, the Certificate Issuer and the ZKP Verifier, and three repositories, one to store the mappings with other service provider ontologies, one to store the sets of synonymns Synsets, and one to store identity verification policies. The Request Manager component handles clients's requests and asks clients the identity attributes necessary for identity verification. The ZKP Verifier performs the AgZKPK verification. The Heterogeneity Management Services provides several functions shared by all CSPs. It consists of two modules: Synset SetUp and Ontology Manager. Synset SetUp returns the set of synonyms of a given term by querying a local thesaurus

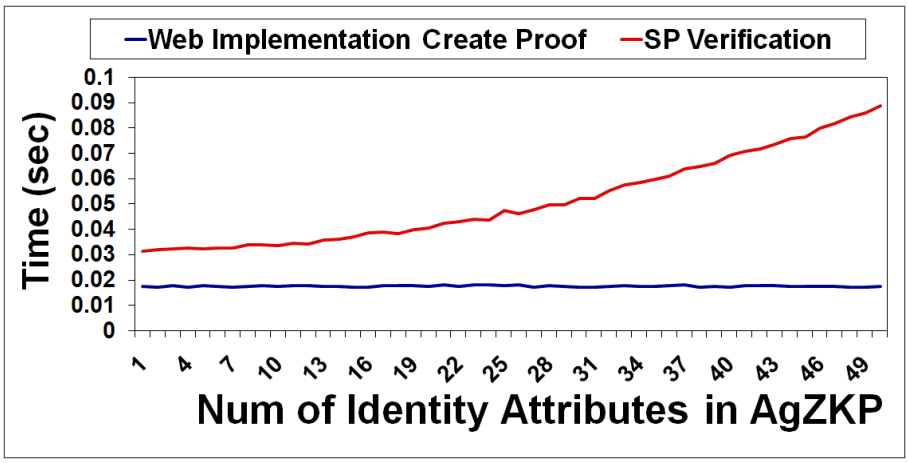

Figure 2: AgZKPK Verification versus Creation

while Ontology Manager provides the functionalities to perform the mapping between two ontologies.

The Service Provider application has been developed in JAVA. It implements the identity attribute name matching protocol using the Falcon-AO v0.7 [2, 3] ontology mapping API and WordNet 2.1 English Lexical database [6]. The User application has been implemented in JSP while the Registrar has been implemented as a JAVA servlet. Finally, we have used Oracle 10g DBMS to store clients' identity records, ontology mappings, set of synonyms, session data, and mapping certificates.

5 Experimental Evaluation

We have performed several experiments to evaluate the AgZKPK protocol that characterizes our approach to multi-factor identity verification and the identity attribute names matching process. We have carried out the following experimental evaluations:

- we have measured the time taken by the Client to generate the aggregate ZKP by varying the number of identity attributes being aggregated from 1 to 50;
- we have measured the time taken by the cloud service for aggregate ZKP verification execution time varying the number of identity attributes being aggregated from 1 to 50 (see Figure 2).

The execution time has been measured in CPU time (milliseconds). Moreover, for each test case we have executed twenty trials, and the average over all the trial execution times has been computed. Figure 2 reports the times to create an AgZKP and to verify it for varying values in the number of identity attributes being aggregated. The execution time to generate the AgZKP (represented by the blue line in the graph) is almost constant for increasing values in the number of identity attributes. The reason is that the creation of AgZKP only requires a constant number of exponentiations. By contrast, the time that the component Web service takes to perform identity attributes verification linearly increases with the number of identity attributes to be verified. The reason is that during the verification the component Web service is required to multiply all the commitments to verify the resulting aggregate signature.

6 Concluding Remarks

In this paper we have proposed an approach to the verification of digital identity for cloud platforms. Our approach uses efficient cryptographic protocols and matching techniques to address heterogeneous naming. We

plan to extend this work in several directions. The first direction is to investigate the delegation of identity attributes from clients to CSPs. Delegation would allow a CSP, called the source CSP, to invoke the services of another CSP, called the receiving CSP, by passing to it the identity attributes of the client. However the receiving CSP must be able to verify such identity attributes in case it does not trust the source CSP. One possibility would be to allow the receiving CSP to directly interact with the client; however the source CSP may not be willing to allow the client to know the CSPs it uses for offering its services. Therefore protocols are needed able to address three requirements: confidentiality of business relations among the various CSPs, user privacy, and strenght of identity verification. The second direction is the investigation of unlinkability techniques. Our approach does not require that the values of the identity attributes only used for identity verification be disclosed to the CSPs; also our approach allows the user to use pseudonyms when interacting with the CSPs, if the CSP policies allow the use of pseudonyms and the user is interested in preserving his/her anonymity. However, if multiple transactions are carried out by the same user with the same CSP, this CSP can determine that they are from the same user, even if the CSP does not know who this user is nor the identity attributes of the user. Different CSPs may also collude and determine a profile of the transactions carried out by the same user. Such information when combined with other available information about the user may lead to disclosing the actual user identity or the values of some of his/her identity attributes, thus leading to privacy breaches. We plan to address this problem by investigating techniques that maintain unlinkability among multiple transactions carried out by the same user with the same or different CSPs.

7 Acknowledgments

This material is based in part upon work supported by the National Science Foundation under the ITR Grant No. 0428554 “The Design and Use of Digital Identities”, upon work supported by AFOSR grant A9550-08-1-0260, and upon work supported by the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001, under the auspices of the Institute for Information Infrastructure Protection (I3P) research program. The I3P is managed by Dartmouth College. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security, the I3P, or Dartmouth College.

References

[1] Bhargav-Spantzel, A., Squicciarini, A.C, Bertino, E.: Establishing and Protecting Digital Identity in Federation Systems. Journal of Computer Security, IOSPress, 14(3), pp. 269–300, (2006)

[2] Choi, N., Song, I. Y., Han, H.: A survey on ontology mapping. SIGMOD Record 35, (3), pp. 34–41.

[3] Falcon, http://iws.seu.edu.cn/projects/matching/

[4] Y. Kalfoglou, and M. Schorlemmer. ”Ontology mapping: the state of the art.” The Knowledge Engineering Review, 18(1), pp. 1–31, (2003).

[5] Pedersen, T.P.: Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. Advances in Cryptology, Proc. CRYPTO ’91, pp. 129–140, (1991).

[6] WordNet, http://wordnet.princeton.edu/