# A Federated Cloud Identity Broker-Model for Enhanced Privacy via Proxy Re-Encryption

Bernd Zwattendorfer, Daniel Slamanig, Klaus Stranacher, Felix Hörandner

# A Federated Cloud Identity Broker-Model for Enhanced Privacy via Proxy Re-Encryption

Bernd Zwattendorfer, Daniel Slamanig*, Klaus Stranacher and Felix Hörandner

Institute for Applied Information Processing and Communications (IAIK),
Graz University of Technology (TUG), Inffeldgasse 16a, 8010 Graz, Austria
{bernd.zwattendorfer,daniel.slamanig,klaus.stranacher}@iaik.tugraz.at
felix.hoerandner@student.tugraz.at

**Abstract.** Reliable and secure user identification and authentication are key enablers for regulating access to protected online services. Since cloud computing gains more and more importance, identification and authentication in and across clouds play an increasing role in this domain too. Currently, existing web identity management models are often just mapped to the cloud domain. Besides, within recent years several cloud identity management models such as the *cloud identity broker-model* have emerged. In the aforementioned model, an identity broker in the cloud acts as hub between various service and identity providers. While this seems to be a promising approach for adopting identity management in cloud computing, still some problems can be identified. A notable issue is the dependency of users and service providers on the same central broker for identification and authentication processes. Additionally, letting an identity broker store or process sensitive data such as identity information in the cloud brings up new issues, in particular with respect to user's privacy. To overcome these problems, we propose a new cloud identity management model based on the federation between different cloud identity brokers. Thereby, users and service providers can select their favorite cloud identity broker without being dependent on one and the same broker. Moreover, it enhances user's privacy by the use of appropriate cryptographic mechanisms and in particular proxy re-encryption. Besides introducing the model we also provide a proof of concept implementation thereof.

**Keywords:** cloud computing, identity management, cloud identity, cloud identity broker, federated cloud identity broker, privacy, proxy re-encryption

## 1 Introduction

In security-sensitive areas of applications such as e-Government identity management is a key issue. Over the time, several identity management systems have already evolved [2]. The Security Assertion Markup Language (SAML),

* Daniel Slamanig has been supported by the Austrian Research Promotion Agency (FFG) through project ARCHISTAR, grant agreement number 832145.

Shibboleth, OpenID, or WS-Federation are just a few popular examples. They all usually follow a similar architectural concept involving the stakeholders *user (U)*, *service provider (SP)*, and *identity provider (IdP)* [3]. Thereby, a user wants to access a protected resource at a service provider. To mitigate efforts for the service provider, the identification and authentication process is handled by the identity provider. After successful authentication, the identity provider transfers identity and user data to the service provider for access decision making.

Since cloud computing plays a steadily increasing role in the IT sector, secure identity management is equally important in the cloud domain. Identity management systems in the cloud can benefit from cloud advantages such as high scalability or cost savings, since no in-house infrastructure needs to be hosted and maintained. A couple of cloud identity management-systems have already evolved [7–9]. One example is the so-called *cloud identity broker-model*, where an identity broker in the cloud acts as hub between multiple service providers and identity providers [7]. The advantage of adopting the broker concept is that the identity broker hides the complexity of different identity providers from the service provider. Although the cloud identity-broker model is a promising model in the cloud domain, still some disadvantages can be found. One major drawback is that both users and service providers have to rely on one and the same cloud identity provider for identification and authentication. This heavily decreases user's and service provider's flexibility in choosing their cloud identity broker of choice. In addition, the *cloud identity broker-model* – when applied in a public cloud – lacks in privacy, because identity data are stored and processed in the cloud. However, privacy is one main issue with respect to cloud computing [15].

To eliminate these problems, we propose a new cloud identity management-model which, on the one hand, increases freedom of choice in terms of cloud identity broker selection and, on the other hand, preserves user's privacy with respect to the cloud identity broker and – whenever possible – to the identity provider. We address the first issue by applying a federation of cloud identity brokers and the second issue by incorporating appropriate cryptographic techniques.

## 2 Federated Cloud Identity Broker-Model

In this section we propose our new cloud identity management-model which federates cloud identity brokers. The general idea is that users encrypt their identity data using their public key of a proxy re-encryption scheme and these data can be re-encrypted to a service provider.

### 2.1 Cryptographic Preliminaries

Subsequently, we briefly discuss required cryptographic primitives and we denote a proxy re-encryption and signature key pair of $A$ by $(\mathsf{sk_A}, \mathsf{pk_A})$ and $(\mathsf{sk'_A}, \mathsf{pk'_A})$ respectively. Concatenation of two bitstrings $a$ and $b$ denoted as $a \| b$ is assumed to be realized in a way such that all individual components are uniquely recoverable.

**Digital Signatures:** A digital signature scheme (DSS) is a triple $(\mathcal{K}, \mathcal{S}, \mathcal{V})$ of poly-time algorithms, whereas $\mathcal{K}$ is a probabilistic key generation algorithm that takes a security parameter $\kappa$ and outputs a private and public key pair $(\mathsf{sk}, \mathsf{pk})$. The probabilistic signing algorithm $\mathcal{S}$ takes as input a message $M \in \{0,1\}^*$ and a private key $\mathsf{sk}$, and outputs a signature $\sigma$. The verification algorithm $\mathcal{V}$ takes as input a signature $\sigma$, a message $M \in \{0,1\}^*$ and a public key $\mathsf{pk}$, and outputs a single bit $b \in \{\mathtt{true}, \mathtt{false}\}$ indicating whether $\sigma$ is a valid signature for $M$.

**Proxy Re-Encryption:** A unidirectional single-use proxy re-encryption (US-PRE) scheme allows a semi-trusted proxy given a re-encryption key to transform a message encrypted under the key of party $A$ into another ciphertext to the same message encrypted for party $B$. The proxy thereby neither gets access to the plaintext nor the respective decryption keys and can only transform in one direction (from $A$ to $B$) and one ciphertext can be transformed only once (no transitivity). A US-PRE is a tuple $(\mathcal{S}, \mathcal{K}, \mathcal{RK}, \mathcal{E}_\mathcal{R}, \mathcal{RE}, \mathcal{D}_\mathcal{R})$ of poly-time algorithms. The algorithm $\mathcal{S}$ runs a setup and produces system parameters $\mathsf{params}$. $\mathcal{K}$ is a probabilistic key generation algorithm that takes a security parameter $\kappa$ and outputs a private and public key pair $(\mathsf{sk_i}, \mathsf{pk_i})$. The re-encryption key generation algorithm $\mathcal{RK}$ takes as input a private key $\mathsf{sk_i}$ and another public key $\mathsf{pk_j}$, and outputs a re-encryption key $\mathsf{rk_{i \to j}}$. The probabilistic encryption algorithm $\mathcal{E}_\mathcal{R}$ gets a public key $\mathsf{pk_i}$ and a plaintext $M$, and outputs $c_i = \mathcal{E}_\mathcal{R}(\mathsf{pk_i}, M)$. The (probabilistic) re-encryption algorithm $\mathcal{RE}$ gets as input a ciphertext $c_i$ under $\mathsf{pk_i}$ and a re-encryption key $\mathsf{rk_{i \to j}}$, and outputs a re-encrypted ciphertext $c_j = \mathcal{RE}(\mathsf{rk}_{i \to j}, c_i)$ for $\mathsf{pk_j}$. The decryption algorithm $\mathcal{D}_\mathcal{R}$ takes private key $\mathsf{sk_j}$ and a ciphertext $c_j$, and outputs $M = \mathcal{D}_\mathcal{R}(\mathsf{sk_j}, c_j)$ or an error $\perp$. We base our implementation on the schemes of [1].

## 2.2 Model Architecture

The proposed new cloud identity management model relies on a federated approach. Thereby, dependency on one single cloud identity broker is removed by using multiple cloud identity brokers that are able to communicate with each other. Users and service providers can select their preferred cloud identity broker for authentication, thus both identity brokers can actually provide and support different functionality. The only prerequisite is that identity data transfer is possible between the individual cloud identity brokers. Figure 1 illustrates this *federated cloud identity broker-model.*

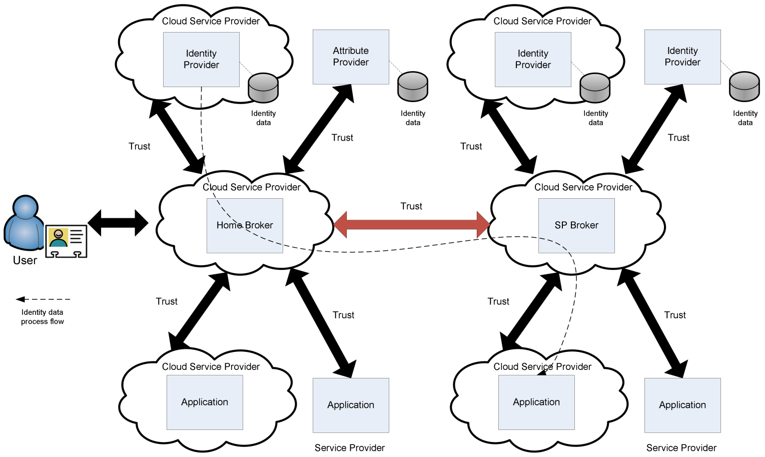

**Fig. 1.** Federated Cloud Identity Broker-Model

In the following, we briefly describe the components involved:

**User:** A user wants to access protected resources from a service provider. For identification and authentication, the user relies on her favorite cloud identity broker (user's home broker), which manages different identity providers and attribute providers the user is registered with.

**Service provider:** A service provider offers various services to users and requires proper identification and authentication.

**Identity provider:** The identity provider stores user's identity data. Furthermore, the identity provider is responsible for user identification and authentication.

**Attribute provider:** The attribute provider stores additional attributes of the user's identity data. These additional attributes can be retrieved from the attribute provider during an authentication process.

**Home broker:** The user's home broker constitutes the cloud identity broker the user is affiliated with. The user trusts this broker and has a contractual relationship with it. The home broker manages all identity providers and attribute providers, where the user is registered with.

**Service provider broker:** The service provider broker (SP broker) has an affiliation with the service provider the user wants to authenticate. The SP broker manages the communication with the user's home broker for the service provider.

### 2.3 Requirements

When designing this new *federated cloud identity broker-model*, we kept the following requirements in mind, which need to be fulfilled:

**Individual selection of the cloud identity broker:** Both users and service providers are able to individually select the cloud identity broker of their choice.

**Trust:** The service provider and identity provider are trusted, whereas the cloud provider which hosts and operates the identity broker, is assumed to be semi-trusted (*honest but curious*). This means, the identity broker works correctly, but might be interested in inspecting users' identity data. With our model we can also assume the identity providers to be semi-trusted.

**Privacy:** For our model we demand the support of the privacy characteristics *user-centricity* (the user always stays under full control on which data are disclosed to the service provider and cloud identity broker) and *selective disclosure* (the user is able to select the amount of data to disclose to the service provider and cloud identity broker). Furthermore, users' identity data should be treated confidential and users' privacy must be preserved with respect to all entities in the cloud.

**Easy integration into existing infrastructures:** The new model should be easily integrable into existing infrastructures, meaning that service providers and identity providers can easily connect to the cloud identity broker through standardized and already existing interfaces.

## 3 Concrete Model and Proof of Concept

Subsequently, we provide details of the model by means of a proof of concept implementation. Thereby, we designed and developed one demo service provider, two cloud identity brokers (the user's home broker and the SP broker), one attribute provider, and additionally integrated two existing identity providers, i.e., Twitter and one self-hosted OpenID provider. Figure 2 illustrates the implemented architecture and its components, which will be described in detail in the next subsection.

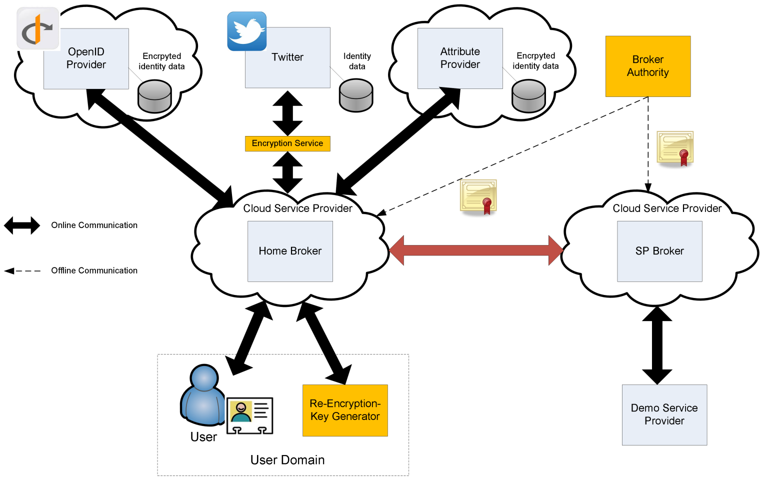

**Fig. 2.** Implementation Architecture of the Federated Cloud Identity Broker-Model

In order to meet the previously defined requirements, three additional components need to be introduced. These new as well as the other components will be described in detail in the next subsection.

### 3.1 Components

In this section we give implementation details on the individual components.

**Demo service provider:** The demo service provider has actually no particular functionality, it just requires proper user identification and authentication. To minimize the amount of data transferred and to respect user's privacy, the service provider is able to request only specific attributes from the user for service provisioning. In addition, the service provider can request a certain level of quality for the identity and the authentication process. This form of quality assurance is modeled as authentication levels, similar to the ones proposed by the NIST [16], STORK [10], or ISO/IEC [12].

**SP broker:** The SP broker has been selected by the service provider and thus they share a contractual relationship. The SP broker communicates with the user's home broker and forwards the authentication request to it. Additionally, the SP broker offers a user interface where the user can provide location information of her home broker.

**Home broker:** The location of the user's home broker is identified via a user-specific URL, which points to this broker. The URL format is similar to the one used by the OpenID protocol. The user-customized URL is not persistent and can be changed by the user anytime. Before being able to use the functionality of the home broker, the user has to register with it. The home broker holds metadata for the user which include the identity providers the user is able to use and is registered with, and which attribute providers can be connected. The home broker communicates with the identity providers for user identification and authentication and with the attribute providers for attribute transfer. During the authentication process, the home broker presents the user an identity provider selection page and the requested attributes from the service provider. Thereby, the user can select the identity source the requested attributes should be retrieved from. If data are retrieved from different identity data sources (e.g., from an identity provider and an attribute provider), the home broker does a mapping to a common (semantic) format.

**Broker authority:** The broker authority is responsible for managing the trust relationships between cloud identity brokers. For that, it issues certificates for signature public keys of the individual brokers. The respective signing keys are used to sign messages exchanged between brokers, ensure an authentic communication channel, and thus verify the trust relationships. Note that this is merely a virtual entity and any (set of) mutually trusted certification authorities will be sufficient in practice.

**Twitter:** In our scenario, we use Twitter as an identity provider. When registering, Twitter stores a couple of user attributes such as the user's full name or language.

**OpenID provider:** In this implementation we set up our own OpenID provider. The reason is that we want to ensure confidentiality of user's attributes with respect to the identity provider and the two brokers. To achieve this, the user encrypts her attributes under the user's public key of a proxy re-encryption scheme before storing them at the OpenID provider. At this stage, only the user is able to decrypt the attributes again. The sole attribute, which is visible in plaintext to the OpenID provider, is the user's OpenID identifier. A similar approach is discussed in [14], however, in this paper data are not encrypted by the user but by a host organization.

**Attribute provider:** For the attribute provider we use the same approach as for the OpenID provider. Hence, the user stores her identity data at the attribute provider in encrypted format only. The only attribute the attribute provider is able to inspect in plaintext is an identifier to link the encrypted attributes to a specific user. At the attribute provider, no explicit user authentication is required.

**Re-Encryption-Key Generator:** The re-encryption-key generator is an entity that runs directly in the user's domain to avoid any private key transfer to another party. In our implementation, the user allows her identity data, which are encrypted for her and stored at the identity/attribute provider, to be re-encrypted by the home broker for a service provider. This way, the identity data remains always confidential even if routed through the identity brokers residing in the cloud. The functionality of the re-encryption-key generator is computing the re-encryption $\mathsf{rk}_{\mathsf{U}\to\mathsf{SP}}$ by taking the private key of the user $\mathsf{sk}_{\mathsf{U}}$ and the public key of the service provider $\mathsf{pk}_{\mathsf{SP}}$.

**Encryption Service:** The encryption service enables the encryption of data coming from an identity provider such as Twitter, which does not support storage of encrypted attributes, by the user. Hence, identity data stays always confidential before transmission to the cloud identity brokers.

### 3.2 Communication Interfaces

We now briefly describe the used communication protocols and how they were implemented. We thereby describe the interfaces and protocols, respectively, between two entities at a time. All communication interfaces are secured using SSL/TLS for transport security, hence this fact will not be mentioned again explicitly in the individual descriptions.

**Service provider ↔ SP broker:** Actually, arbitrary identity and authentication protocols can be used for this communication channel. Nevertheless, in our implementation we relied on an amended version of the SAML AuthnRequest/Response Protocol [5] using the SAML HTTP-POST Binding [4]. In particular, amendments are the inclusion of requested attributes as well as the requested authentication level in the SAML authentication request. In fact, the amended protocol is similar to the STORK protocol [11], which will play an important role in identification and authentication processes across Europe in the near future[1]. Trust is established by means of signature certificates. However, there is no explicit trust framework required, trust can be negotiated bilaterally.

**SP broker ↔ home broker:** Again, for this communication path we rely on the amended SAML protocol. Exchanged messages are also digitally signed (certificates are signed by the trusted broker authority). This ensures that only by the authority authorized brokers are able to trust and communicate with each other.

**Home broker ↔ Twitter:** For retrieving identity data from Twitter we used the OAuth 1.0 protocol. However, the communication path is intercepted by the trusted encryption service that allows users to encrypt their identity data before presenting it to the home broker.

[1] There are only minor differences between our used SAML protocol and the STORK protocol. Differences mainly target the format and semantic of transferred attributes, as e.g., single encrypted attributes are not supported within STORK.

**Home broker ↔ OpenID provider:** For this communication channel we implemented the OpenID 2.0 interface. This is somewhat related to the work in [14].

**Home broker ↔ attribute provider:** For simplicity, in our proof of concept implementation we use a customized web service interface. The request message includes requested attributes and an identifier of the user, the response then simply returns the corresponding encrypted attributes.

**Home broker ↔ re-encryption-key generator:** Communication is based on the SAML AttributeQuery/Response Protocol [5]. The attribute query thereby includes the public key of the service provider $\mathsf{pk}_{\mathsf{SP}}$. By calling the local re-encryption-key generator with the users private key $\mathsf{sk}_{\mathsf{U}}$ the user obtains the re-encryption key $\mathsf{rk}_{\mathsf{U}\rightarrow\mathsf{SP}}$, which is wrapped in the response. In our implementation we use a non-interactive, unidirectional, and single-use proxy re-encryption scheme of [1].

**Broker authority ↔ SP broker/home broker:** The exchange of certificates between the broker authority and the brokers is actually an offline process. Exchange is carried out using appropriate organizational mechanisms.

### 3.3 Process Flows

Subsequently, we present the secure identification and authentication process using the implementation of our proposed *federated cloud identity broker-model*. Identification and authentication is explained by contacting the OpenID and the attribute provider.

**Setup:** The following setup is required before running an authentication process:

- We assume the user trusts the service provider, Twitter, the encryption service, and the re-encryption-key generator (latter runs in the user's domain). In contrast to that, we assume the cloud identity brokers (SP broker and home broker), the OpenID provider, and the attribute provider semi-trusted (*honest but curious*), meaning that they work correctly but might be interested in inspecting user's data.
- The broker authority has certified the trustworthiness of the two brokers by certifying the signature public keys and thus verifying the trust relationship between the brokers. We denote the respective signature key pairs as $(\mathsf{sk}'_{\mathsf{SP-Broker}}, \mathsf{pk}'_{\mathsf{SP-Broker}})$ and $(\mathsf{sk}'_{\mathsf{Home-Broker}}, \mathsf{pk}'_{\mathsf{Home-Broker}})$. These keys are used for signing the SAML messages exchanged between the two brokers.
- A bilateral trust relationship has been negotiated between the service provider and the SP broker. To enforce this trust relationship on technical level, certified signature public keys have been exchanged. We denote these signing key pairs of the SP $(\mathsf{sk}'_{\mathsf{SP}}, \mathsf{pk}'_{\mathsf{SP}})$ and assume that the SP broker uses $(\mathsf{sk}'_{\mathsf{SP-Broker}}, \mathsf{pk}'_{\mathsf{SP-Broker}})$. These keys are used for signing the exchanged SAML messages between SP and SP Broker. In addition, the service provider holds a proxy re-encryption key pair $(\mathsf{sk}_{\mathsf{SP}}, \mathsf{pk}_{\mathsf{SP}})$.

- A bilateral trust relationship exists between the user's home broker and the individual identity providers. The establishment of this trust relationship is protocol dependent, however, both channels (between home broker and Twitter and between home broker and the OpenID provider) are authentic.
- The user possesses a proxy re-encryption key pair $(\mathsf{sk_U}, \mathsf{pk_U})$ and has already stored personal attributes in encrypted format at the OpenID provider and the attribute provider. We denote a set of user encrypted attributes as $c_{U_i} = (c_{U_1}, \ldots, c_{U_m})$ and the corresponding plaintext attributes as $a_i = (a_1, \ldots, a_m)$.
- The user has a contractual relationship with the home broker, has registered in her profile the identity/attribute providers she wants to use, and has stored appropriate authentication credentials for the attribute provider. Additionally, the user holds a unique personal identifier ($\mathsf{uniqueID}$) to be identifiable at the home broker.

**Authentication Process:**

1. A user wants to access a protected resource from the service provider.
2. Since the service provider requires authentication, it forwards the user to its affiliated SP broker. This SAML authentication request includes the set of attributes ($\mathsf{req\_attr}$), which should be provided during the authentication process, the requested authentication level ($\mathsf{req\_auth\_level}$), and the public encryption key $\mathsf{pk_{SP}}$ of the service provider. The request is signed by the service provider resulting in signature $\sigma_{SP} = \mathcal{S}(\mathsf{sk'_{SP}}, \mathsf{req\_attr} \| \mathsf{req\_auth\_level} \| \mathsf{pk_{SP}})$
3. First, the broker verifies $\sigma_{SP}$. Furthermore, the SP broker asks the user to provide location information of her home broker. The user enters a URL, which is a composition of a $\mathsf{uniqueID}$ of the user at the home broker and the home broker's domain (e.g., `https://user.home-broker.com`).
4. The SP broker again creates a signature $\sigma_{SP-Broker} = \mathcal{S}(\mathsf{sk'_{SP-Broker}}, \mathsf{req\_attr} \| \mathsf{req\_auth\_level} \| \mathsf{pk_{SP}} \| \mathsf{uniqueID})$ and forwards the authentication request of the SP to the user's home broker (using the SAML protocol).
5. The home broker verifies $\sigma_{SP-Broker}$. Based on the $\mathsf{uniqueID}$, the user is identified at the home broker. The home broker presents the user a web page, which shows the requested attributes $\mathsf{req\_attr}$ of the service provider. Additionally, the user can select at which identity provider she wants to authenticate (only those identity providers are shown, which were registered by the user and which support the requested authentication level $\mathsf{req\_auth\_level}$). Furthermore, the user can select for every individual attribute if it should be retrieved from the identity provider – if providable – or from an affiliated attribute provider. In our example we assume that the user selects the OpenID provider as an identity provider and that additional attributes should be retrieved from the attribute provider.
6. Based on the user's OpenID identifier the user is redirected to the OpenID provider.
7. The user authenticates at the OpenID provider using appropriate credentials.

8. The attributes, which have been selected for retrieval from the OpenID provider, are returned to the home broker in encrypted fashion. We assume the user encrypted attributes $(c_{U_1}, \dots, c_{U_j})$ to be returned.
9. Since in our scenario only a subset of the requested attributes can be retrieved from the OpenID provider, additional attributes are fetched from the attribute provider. Communication and retrieval is based on a pre-negotiated access token as used in OAuth, which is shared between the home broker and the attribute provider, to identify the user at the attribute provider and allow the broker access to the user's data.
10. The remaining attributes $(c_{U_k}, \dots, c_{U_m})$ are returned to the home broker in encrypted format.
11. Now all requested attributes $(c_{U_1}, \dots, c_{U_m})$ are located at the home broker, but they are still encrypted for the user. To make these attributes readable for the SP, re-encryption needs to be applied. A re-encryption key generation request is sent by the home broker to the local re-encryption key generator, which includes the public key of the service provider $\mathsf{pk}_{\mathsf{SP}}$. The user additionally has to provide the key generator access to her private key $\mathsf{sk}_{\mathsf{U}}$.
12. The re-encryption key generator computes the re-encryption key from the service provider's public and the user's private key and returns the re-encryption key $\mathsf{rk}_{\mathsf{U}\rightarrow\mathsf{SP}} = \mathcal{RK}(\mathsf{sk}_{\mathsf{U}}, \mathsf{pk}_{\mathsf{SP}})$ to the home broker.
13. The home broker re-encrypts all collected attributes for the service provider resulting in $(c_{SP_1} \dots, c_{SP_m})$ by running $c_{SP_i} = \mathcal{RE}(\mathsf{rk}_{U\rightarrow SP}, c_{U_i})$ for all $1 \leq i \leq m$. Additionally, it wraps the re-encrypted attributes and the actual authentication level auth_level into a SAML assertion and computes a signature $\sigma_{Home-Broker} = \mathcal{S}(\mathsf{sk}'_{\mathsf{Home-Broker}}, \mathsf{c}_{\mathsf{SP}_1} \| \dots \| \mathsf{c}_{\mathsf{SP}_\mathsf{m}} \| \mathsf{auth\_level})$.
14. The SAML assertion is returned within the authentication response to the SP broker. The SP broker verifies $\sigma_{Home-Broker}$, computes a signature $\sigma_{SP-Broker} = \mathcal{S}(\mathsf{sk}'_{\mathsf{SP-Broker}}, \mathsf{c}_{\mathsf{SP}_1} \| \dots \| \mathsf{c}_{\mathsf{SP}_\mathsf{m}} \| \mathsf{auth\_level})$ and forwards the authentication response to the service provider.
15. The service provider verifies the received response by verifying $\sigma_{SP-Broker}$ and obtains the decrypted attributes $(a_1 \dots, a_m)$ by running $a_i = \mathcal{D}_{\mathcal{R}}(\mathsf{sk}_{\mathsf{SP}}, \mathsf{c}_{\mathsf{SP}_\mathsf{i}})$ for all $1 \leq i \leq m$.
16. Based on the decrypted identity and attribute data $(a_1 \dots, a_m)$ and the auth_level the service provider is able to provide the desired protected resources to the user.

In contrast to the above description, Twitter does not allow to store encrypted data. However, we still are able to achieve privacy when using Twitter. In this case, identity data needs to be encrypted by the user before being transferred from Twitter to the home broker.

**Recurring Authentications:** Most of the time, running through the complete authentication process described before might be cumbersome for the user. Therefore, our implementation is able to remember some selections the user did in her first authentication process, if the user wants so. For instance, in a recurring authentication process the steps 3 and 4 (indicating the home broker) can be omitted, because the SP broker is able to remember user's choice during her first authentication. In addition, step 9 (providing authentication credentials to the identity provider) can be skipped if single sign-on (SSO) [6] is supported by the selected identity provider. Also the key generation steps 13-15 are not necessary, as the re-encryption key for a particular service provider can be stored for re-use in the user's profile at the home broker. Avoiding as many user interactions as possible definitely increases usability of our solution.

## 4 Evaluation and Discussion

In this section we evaluate our model and implemented solution regarding the requirements specified in Section 2.3.

**Individual selection of the cloud identity broker:** Both, the user and the service provider are able to select the cloud identity broker of their choice. The service provider just needs to establish a trust relationship with the broker and implement the communication interface it offers. In addition, the user can contract another broker and registers her desired identity and attribute providers. The user is identified by the broker by a uniqueID.

**Trust:** Trust between two broker is grounded through the broker authority. The pairwise trust relationships between service provider and SP broker, and between home broker and identity provider depend on bilateral agreements. There is no direct trust relationship between service provider and identity provider because the brokers act as intermediary. Hence, trust is brokered between service provider and identity provider.

**Privacy:** The requirement of user-centricity is achieved because individual attributes can be stored encrypted for the user only at an identity provider or attribute provider. If this is not possible (e.g., with Twitter), a trusted encryption service can be used as intermediary to encrypt identity data before transmitting it to the cloud identity broker. Only the user is in control to decrypt the data or to generate re-encryption keys. We support selective disclosure because the user is able to select the attributes she wants to transfer at the home broker (i.e. the service provider only gets the attributes which it has requested and the user gave consent for). In addition, confidentiality of user attributes with respect to the cloud identity broker is achieved through proxy re-encryption.

**Easy integration into existing infrastructures:** The complete model can be easily adopted by service providers. Service providers just need to establish a contractual and trust relationship with their desired SP broker. Furthermore, they just need to implement one specific interface to the SP broker and not many interfaces to different identity providers as required in traditional settings. Implementation efforts can be reduced by providing appropriate software libraries. Additional identity providers or attribute providers can be easily integrated by home brokers. The brokers just need to implement their communication protocols offered by the identity providers or attribute providers.

## 5 Conclusions and Future Work

In our proof of concept implementation we showed that federating identity brokers provides greater flexibility to users in identity/attribute provider selection. However, such a brokered trust relationship might bring up liability discussions, in particular, if identity providers are grounded by national law.

Besides setting up a more sophisticated and complex network of cloud identity brokers, future work will include the integration of additional providers such as Facebook, Google, or even national eID solutions (e.g., based on ideas related to [17]). Moreover, the integration of the STORK framework [13] could boost the number of (high quality) identity providers supported. A possible approach, how this could be realized, has been discussed in [18].

## References

1. Ateniese, G., Fu, K., Green, M., Hohenberger, S.: Improved proxy re-encryption schemes with appl. to secure distributed storage. ACM Trans. Inf. Syst. Secur. 9(1), 1–30 (2006)
2. Bauer, M., Meints, M., Hansen, M.: D3.1: Structured Overview on Prototypes and Concepts of Identity Management System. FIDIS (2005)
3. Bertino, E., Takahashi, K.: Identity Management: Concepts, Technologies, and Systems. Artech House (2011)
4. Cantor, S., Hirsch, F., Kemp, J., Philpott, R., Maler, E.: Bindings for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS (2009)
5. Cantor, S., Kemp, J., Philpott, R., Maler, E.: Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS (2009)
6. Clercq, J.D.: Single sign-on architectures. In: InfraSec. pp. 40–58 (2002)
7. Cloud Security Alliance: SECURITY GUIDANCE FOR CRITICAL AREAS OF FOCUS IN CLOUD COMPUTING V3.0. CSA (2011)
8. Gopalakrishnan, A.: Cloud Computing Identity Management. SETLabs Briefings 7(7), 45–55 (2009)
9. Goulding, J.T.: identity and access management for the cloud : CA Technologies strategy and vision. Tech. Rep. May, CA Technologies (2010)
10. Hulsebosch, B., Lenzini, G., Eertink, H.: STORK D2.3 - Quality authenticator scheme. Tech. rep., STORK (Mar 2009)
11. J. Alcalde-Morano, e.: STORK D5.8.3b Interface Specification. STORK (2011)
12. JTC1/SC27: ISO/IEC DIS 29115 - Information technology – Security techniques – Entity authentication assurance framework (2013)
13. Leitold, H., Zwattendorfer, B.: STORK: Architecture, Implementation and Pilots. In: ISSE. pp. 131–142 (2010)
14. Nuñez, D., Agudo, I., Lopez, J.: Integrating OpenID with Proxy Re-Encryption to enhance privacy in cloud-based identity services. In: CloudCom. pp. 241 – 248 (2012)
15. Pearson, S., Benameur, A.: Privacy, Security and Trust Issues Arising from Cloud Computing. In: IEEE CloudCom. pp. 693–702 (Nov 2010)
16. William E. Burr et al.: SP 800-63-1. Elec.Authentication Guideline (2011)
17. Zwattendorfer, B., Slamanig, D.: On Privacy-Preserving Ways to Porting the Austrian eID System to the Public Cloud. In: IFIP SEC 2013. pp. 300–314 (2013)
18. Zwattendorfer, B., Slamanig, D.: Privacy-preserving realization of the stork framework in the public cloud. In: SECRYPT. pp. 419–426 (2013)