Felix Hörandner, BSc

A Privacy-Preserving Identity Broker in the Cloud Employing Proxy Re-Encryption with Keys Managed by a CrySIL Service

MASTER'S THESIS

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Computer Science

submitted to

Graz University of Technology

Supervisor

O.Univ.-Prof. Dipl.-Ing. Dr.techn. Reinhard Posch

Institute for Applied Information Processing and Communications (IAIK)

Graz, April 2016

AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis dissertation.

SIGNATURE INFORMATION	**Signatory**	Felix Hörandner
	Date/Time-UTC	2016-04-06T11:45:57+02:00
	Verification	Information about the verification of the electronic signature can be found at: https://www.signaturpruefung.gv.at
Note	This document is signed with a qualified electronic signature. According to § 4 art. 1 of the Signature Act it in principle is legally equivalent to a handwritten signature.	

Date

Signature

Abstract

Identity management plays a crucial role in regulating access to protected resources, as authorization decisions are primarily based on the users' authenticated identities and attributes. An identity broker offers to authenticate users via supported identity providers and presents their attributes to services. This simplifies the services' effort required to perform authentication, while users can choose their preferred identity providers and therefore only have to remember one set of credentials. Proxy re-encryption even allows to exploit the benefits of operating such a broker in the cloud, as this cryptographic primitive enables secure sharing of sensitive data in an untrusted environment. However, the management and on-demand provisioning of involved key material represents a considerable challenge, which has thus far prevented the operational adoption of proxy re-encryption into identity management systems.

This thesis proposes an identity broker system that manages the key material required for proxy re-encryption by embedding user-trusted cryptographic services through a cryptographic service interoperability layer. Consequently, the user controls the sharing of her sensitive data by advising her cryptographic service to issue key material for the broker on demand. An implementation demonstrates the feasibility of the proposed system as well as the particular integration into existing protocols and systems. The included security analysis not only defines general security objectives for the proposed concept, but also evaluates the measures taken in the specific implementation. Also, an overview of multiple proxy re-encryption schemes and their properties is provided, which is then used to select suitable schemes for the proposed identity broker system. Additionally, the impact of emulating the unstandardized proxy re-encryption with traditional encryption schemes is examined.

As a result, this thesis advances the state-of-the-art of identity management by designing and implementing an identity broker system that securely shares the user's attributes in an the cloud, but still offers user-friendly keys management via an interoperable cryptographic service.

Keywords. Identity Broker, Cloud, Proxy Re-Encryption, Cryptographic Service Interoperability Layer

Kurzfassung

Identitätsmanagement spielt eine essentielle Rolle in der Realisierung von Zugangskontrollen auf geschützte Daten, da Authorisierungsentscheidungen primär anhand von authentischen Attributen sowie der Identität einer Benutzerin bzw. eines Benutzers getroffen werden. Hierfür kann ein Identitätsbroker verwendet werden, der AnwenderInnen mit Hilfe von unterstützten Identitätsanbietern für Service Provider authentifiziert und freigegebene Informationen über die BenutzerInnen bereitstellt. Dies reduziert den Aufwand für Service Provider signifikant, während BenutzerInnen ihren jeweils bevorzugten Identitätsanbieter nutzen können, für den sie sich nur die entsprechenden Anmeldedaten merken müssen. Das kryptographische Prinzip der Proxy Re-Encryption ermöglicht es einen Broker in der Cloud zu betreiben ohne in dieser unsicheren Umgebung die Vertraulichkeit von persönlichen Daten zu verletzen, obwohl diese mit ermächtigten Service Providern geteilt werden. Eine erhebliche Herausforderung stellt jedoch die Verwaltung und Ausstellung des benötigten kryptographischen Schlüsselmaterials dar, was bisher die Einbindung von Proxy Re-Encryption in operative Identitätsmanagement-Systeme verhindert hat.

Diese Masterarbeit stellt ein Identitätsmanagement-System vor, das Schlüssel für Proxy Re-Encryption mit Hilfe eines von der Benutzerin bzw. vom Benutzer ausgwählten oder sogar betriebenen kryptographischen Dienstes bereitstellt, welcher über eine Interoperabilitätsschicht angebunden wird. In diesem System kontrollieren die BenutzerInnen die Weitergabe ihrer sensiblen Daten, da nur sie ihre kryptographischen Dienste anweisen können bei Bedarf für den Broker Re-Encryption-Schlüssel auszustellen. Eine Implementierung demonstriert die Umsetzbarkeit des vorgeschlagenen Systems sowie nötige Integrationsschritte für bestehende Protokolle und Systeme. Die durchgeführte Sicherheitsanalyse definiert nicht nur generelle sicherheitsrelevante Zielvorgaben, sondern evaluiert auch die angewendeten Maßnahmen der Implementierung. Des Weiteren wird eine Übersicht der Proxy Re-Encryption Algorithmen und deren Eigenschaften präsentiert, welche die Auswahl von geeigneten Algorithmen für das vorgeschlagene Identitätsbroker-System erlaubt. Zusätzlich wird auch die Emulierung von Proxy Re-Encryption durch traditionelle Verschlüsselungstechniken untersucht und Auswirkungen auf die Anwendbarkeit im geplanten System evaluiert.

Das Ergebnis dieser Arbeit treibt den aktuellsten Stand der Technik im Bereich des Identitätsmanagements voran. Das entworfene und implementierte Identitätsbroker-System ermöglicht es sowohl sensible personenbezogene Daten durch Proxy Re-Encryption in einer unsicheren Cloudumgebung zu teilen, als auch das benötigte kryptografische Schlüsselmaterial komfortabel über einen interoperablen kryptographischen Dienst bereitzustellen.

Schlüsselwörter. Identitätsbroker, Cloud, Proxy Re-Encryption, Interoperabilitätsschicht für kryptographische Dienste

Acknowledgements

I would like to thank Florian Reimair and Peter Teufl, who supported me throughout the research, implementation and writing process of this thesis. Their helpful feedback and effort in correcting draft versions proved to be invaluable.

Additionally, I would like to thank Bernd Zwattendorfer, who introduced me to the field of identity management and motivated me to pursue research in IT security.

Finally, I would like to thank my friends and family for the tremendous support they have been showing the last years.

Felix Hörandner
Graz, April 2016

Contents

List of Figures

List of Tables

1. Introduction

Identity management is an integral requirement to regulate access to protected online services. In order to reach authorization decisions, these services require secure and reliable identification and authentication of their users as well as authentic attributes.

Different models for identity management with increasing sophistication have been introduced to ease the required organizational effort, to reduce the involved complexity of trust relationships, and to improve the user experience. Firstly, in a primitive approach, the service implements the required identity management processes itself, which presents a significant challenge with regards to security. Also, as users should use fresh credentials each service, they have to remember a possibly huge set of different login data. Secondly, by delegating the authentication procedure to a central entity, namely an identity provider, users can reuse one account for a variety of services. However, this model requires both the user as well as the service provider to trust and associate with the same identity provider, which is impractical. Thirdly, a broker acts as a hub between services and identity providers. This broker has to be integrated and trusted by the service, while the user is able to choose from a number of supported identity providers.

The cloud as deployment location for such an identity broker offers attractive features, but the trust implications require advanced cryptography to preserve the user's privacy. When applying a brokered identity management solution to multiple well-used services and a colossal number of authentication processes has to be performed, the cloud's scalability and cost efficiency become very beneficial. However, relinquishing control to a cloud provider, who might inspect the processed data, is especially critical if sensitive identity data is involved. As Nuñez, Agudo, and Lopez [NAL12] as well as Zwattendorfer et al. [ZS13; Zwa+14] have proposed, this challenge can be overcome with an advanced cryptographic primitive called proxy re-encryption. Initially, the user encrypts her attributes for herself. Then, by providing a re-encryption key, the broker can translate these ciphertexts for a service without learning the underlying plaintext. As a result, proxy re-encryption allows to handle and share sensitive data in an untrusted environment.

In order to use such cryptography, key material is required and therefore has to be provisioned and managed. As the user's private key is involved in the generation of a re-encryption key, this complex operation has to be performed by cryptographic software in the user's domain. Furthermore, since users might want to authenticate and share data from various devices, the private key material as well as the operation to generate a re-encryption key have to be independent from the used device. These severe requirements are the reason why previous approaches to integrate proxy re-encryption into an identity management system did not yet offer a solution. As a result, user-friendly key management is still an open question, which prevents the operational adoption of an identity management system employing proxy re-encryption.

This thesis proposes a cloud-based identity broker system that uses a cryptographic interoperability layer to embed user-trusted cryptographic services which manage the involved key material. For example, a cryptographic service can be deployed on a trusted server or on the user's smart phone. Such services only issue re-encryption keys to the broker after the user granted permission. Therefore, the user is able to control the sharing process of her sensitive attributes. Furthermore, as such cryptographic services are independent from the device used for authentication, the user can utilize multiple devices to complete the authentication process, even without the need to install additional software at these devices.

As a result, this thesis expands the forefront of research in the field of identity management by designing and implementing a secure identity broker system, which offers a user-friendly key management solution that is integrated through an interoperability layer to provide the required key material for proxy re-encryption.

1.1. Contribution

This thesis offers five main contributions:

1. **Proposal for an Identity Broker System:** This thesis proposes an identity broker system that uses proxy re-encryption to preserve the user's privacy and manages the involved key material at user-trusted cryptographic services, which are integrated through a cryptographic interoperability layer. In this system, the broker authenticates the users via identity providers, retrieves the required attributes from attribute providers, and re-encrypts these encrypted attributes for service providers with key material obtained from the users' cryptographic services.

2. **Implementation of the Proposed System:** The implementation for such a proposed identity broker system is explained. This implementation also demonstrates how proxy re-encryption can be integrated with existing identity management protocols as well as the cryptographic interoperability layer.

3. **Security Analysis:** Also, this thesis contains a detailed security analysis that consists of two parts. First, generic security objectives are defined, which have to be reached by an identity broker system in order to protect its assets from threats. Second, this analysis examines the implementation's countermeasures against those threats.

4. **Evaluation of Proxy Re-Encryption:** Furthermore, this thesis evaluates the cryptographic primitive proxy re-encryption. This evaluation provides an overview of multiple proxy re-encryption schemes and their properties. Additionally, property requirements for the application in the proposed identity broker system are defined and then used to identify suitable schemes.

5. **Evaluation of Emulated Re-Encryption:** In addition, this thesis explores the emulation of proxy re-encryption implemented by sequentially decrypting and encrypting. This emulated re-encryption is compared to proxy re-encryption and the impact on deployment options inside the proposed identity broker system is examined.

1.2. Outline

This thesis is organized as follows: First, chapter **??** contains an introduction to this thesis. Next, chapter 2 introduces background technologies that represent the basic building blocks. Then, in chapter 3, the proposed identity broker system is described in detail. Chapter 4, provides an overview of proxy re-encryption schemes with their attributes and evaluates schemes regarding their use in the proposed system. In chapter 5, proxy re-encryption is compared to emulated re-encryption, which can be implemented by traditional public key encryption. Subsequently, chapter 6 explains the implementation of the proposed identity broker system. Afterwards, chapter 7 defines security objectives for a generic identity broker system and analyzes the security of the implemented system. In chapter 8, future research directions are proposed. Finally, chapter 9 provides the conclusion of this thesis.

2. Background

This chapter introduces background technologies that build the basis of the proposed identity broker system. It is organized as follows: First, section 2.1 describes the JSON Web Token format as well as related standards published by the JSON Object Signing and Encryption. Then, in section 2.2, the OpenID Connect identity protocol is explained. Section 2.3 examines proxy re-encryption, which allows a proxy to transform a ciphertext for one entity to a ciphertext for another entity, without revealing the underlying plaintext. Finally, section 2.4 describes the cryptographic service interoperability layer, which allows cooperation of multiple nodes in order to fulfill cryptographic requests.

2.1. JSON Web Token (JWT)

JSON Web Tokens (JWTs) are a compact representation of JSON data that are signed and encrypted. JWT is based on the set of four related specifications published by the JSON Object Signing and Encryption (JOSE) working group. Those underlying specifications describe the use of cryptographic operations on JSON data and the representation of the operations' results in JSON. This section first describes the four JOSE specifications, namely JSON Web Algorithms, JSON Web Keys, JSON Web Signature and JSON Web Encryption. Subsequently, further details about JWT are provided.

The JSON Web Algorithms (JWA) standard [Jon15a] defines identifiers and required parameters for cryptographic algorithms. These algorithms are concerned with digital signatures, key management including key encryption and agreement, as well as content encryption. In addition, JWA specifies parameters required for keys and algorithms. The definitions introduced in this standard are used in the accompanying standards.

The JSON Web Key (JWK) standard [Jon15b] specifies a JSON structure for cryptographic key material. Such a structure contains the key type and intended usage, as well as, an algorithm identifier and required parameters defined by JWA. Also, information regarding the key's certificate chain can be included.

The JSON Web Signature (JWS) standard [JBS15a] defines a JSON-based data structure to represent content that is secured with a digital signature or Message Authentication Code (MAC). Such a JWS structure consists of three parts, the JOSE header, the payload, and, finally, the signature. Firstly, the JOSE header specifies the used algorithm with JWA, the key with JWK, and the content type of the payload. Secondly, the payload can be any octet string. Thirdly, the signature value is the output of the signature algorithm or MAC over the header and payload. In order to create such a signature, JWS also defines the transformation process which leads to the canonical input value for the used signature algorithm. The corresponding verification

process is also described. In addition, JWS provides details on how the header, payload and signature can be serialized into either a JSON object or a compact representation intended for space constrained applications.

The JSON Web Encryption (JWE) standard [JH15] defines how encrypted content can be represented using JSON-based data structures. Such a JWE structure consists of four main parts. Firstly, the JOSE header specifies the used algorithms with JWA, the key with JWK, and the type of the encrypted content. Secondly, an encrypted key may be included representing a symmetric content encryption key that was encrypted with an asymmetric encryption scheme. Thirdly, a part is added holding the initialization vector for the encryption algorithm. Finally, the JWE structure contains the output of the encryption algorithm, namely the ciphertext and authentication tag. The JWE standard specifies both an encryption as well as a decryption process. In addition, similar to the definition for JWS, two serialization formats are described, namely a compact format for space limited environments and a serialization into a JSON object.

The JSON Web Token (JWT) standard [JBS15b] uses the four standards by JOSE to define a compact representation of claims that are integrity-protected and/or encrypted. On the one hand, those claims include standardized parameters, such as the issuer of the claims, the subject, the intended audience, and the valid time period. On the other hand, the claims contain application-specific data, for example the user's full name or address. JWT also standardizes the creation process of such a token. The integrity and confidentiality of the claims is protected by JWS and JWE, respectively. If both integrity and confidentiality are desired, this can be achieved by nesting the JWS and JWE structures. The resulting data is serialized into a compact format. In addition to the creation of a token, JWT also specifies the validation process.

In conclusion, JWT allows to protect the integrity and confidentiality of JSON data, while representing it in a compact format. In order to perform cryptographic operations on such JSON data, JWT employs a set of four related standards provided by the JOSE working group. JWA specifies algorithms with their identifiers and required parameters. JWK defines a JSON representation of cryptographic key material. JWS uses these two standards to create a standardized process to sign data and serialize the results. Finally, JWE describes encryption of arbitrary data using algorithms by JWA and keys represented as JWK.

2.2. OpenID Connect

OpenID Connect [Sak+14] is an identity protocol on top of OAuth 2.0 [Har12], which allows service providers (SP) to delegate user authentication to an identity provider (IdP), to obtain data about the user, and to gain authorization to access additional resources. As a result of this process the identity provider issues JSON Web Tokens (JWT) to the service provider, containing identity and attribute information. By offering these data as JWT, confidentiality, integrity and authenticity can be ensured. By delegating the authentication to the user's preferred identity provider, usability improvements can be achieved. The user only has to memorize one set of credentials for her identity provider instead of sets for each service. Furthermore, with single sign-on (SSO), information about the user's recent authentication is stored and can be reused to complete a subsequent authentication request without requiring user interaction. OpenID

Connect supports server-side and client-side web applications as well as mobile apps. Extensions for OpenID Connect also include the automatic discovery of a user's identity provider, dynamic registration of new service providers, and session management.

This section is organized as follows: To begin with, the general authentication process and the involved steps are described. The subsequent paragraphs explain the optional extensions for discovery, dynamic registration, and, finally, session management.

2.2.1. Authentication Process

The authentication process of OpenID Connect involves interaction between user, service and identity provider. In order to access a protected resource at the service provider, the user first has to authenticate via the identity provider. Then, the proof of authentication, the user's attributes, as well as authorization to access further resources have to be transmitted back to the service provider. The service provider can use this information to request further data about the user and, ultimately, reach an access control decision regarding the user's original request. Figure 2.1 illustrates the steps of the OpenID Connect process for the so called authorization code flow. These following paragraphs elaborate on these steps.

1. **Request Protected Resource:** When a user tries to access a protected resource at the service provider, this service provider has to make an authorization decision based on the user's authenticated identity or her attributes. With OpenID Connect, this process of user authentication and attribute acquisition can be delegated to an identity provider. In order to perform such a delegation, the service and identity provider have to be associated, which can be established with extensions for discovery and dynamic registration.

2. **Authenticate at Identity Provider:** The service provider redirects the user's browser to the identity provider with a request for authentication and a list of required permissions. The authentication is performed out of band at the identity provider, which allows the integration of a multitude of authentication methods. The resulting proof of authentication and required attributes are compiled into an id-token. In addition, an access-token representing authorization to access further resources at the identity provider is generated.

3. **Retrieve Tokens:** There are three methods, called flows, that define how these id- and access-tokens can be transmitted back to the service provider. Firstly, in the authorization code flow, the identity provider redirects the user's browser to the service provider. The authentication code is added as GET parameter to the destination URL pointing at the service provider, which then exchanges this code for tokens. This authorization code flow is illustrated in figure 2.1. Secondly, in the implicit flow, the id-token and access-token are sent directly back to the service provider again as parameter in a redirect. Thirdly, in hybrid flows, one token is sent back directly, whereas the other token has to be obtained by exchanging the authorization code.

4. **Request Further Data:** In some scenarios, the service provider requires further data associated with the user. To access these data, the user has to prove her previously obtained authorization. Of particular interest regarding the OpenID Connect process are the OAuth 2.0 Bearer Token [JH12] specification as well as the JSON Web Token profile for

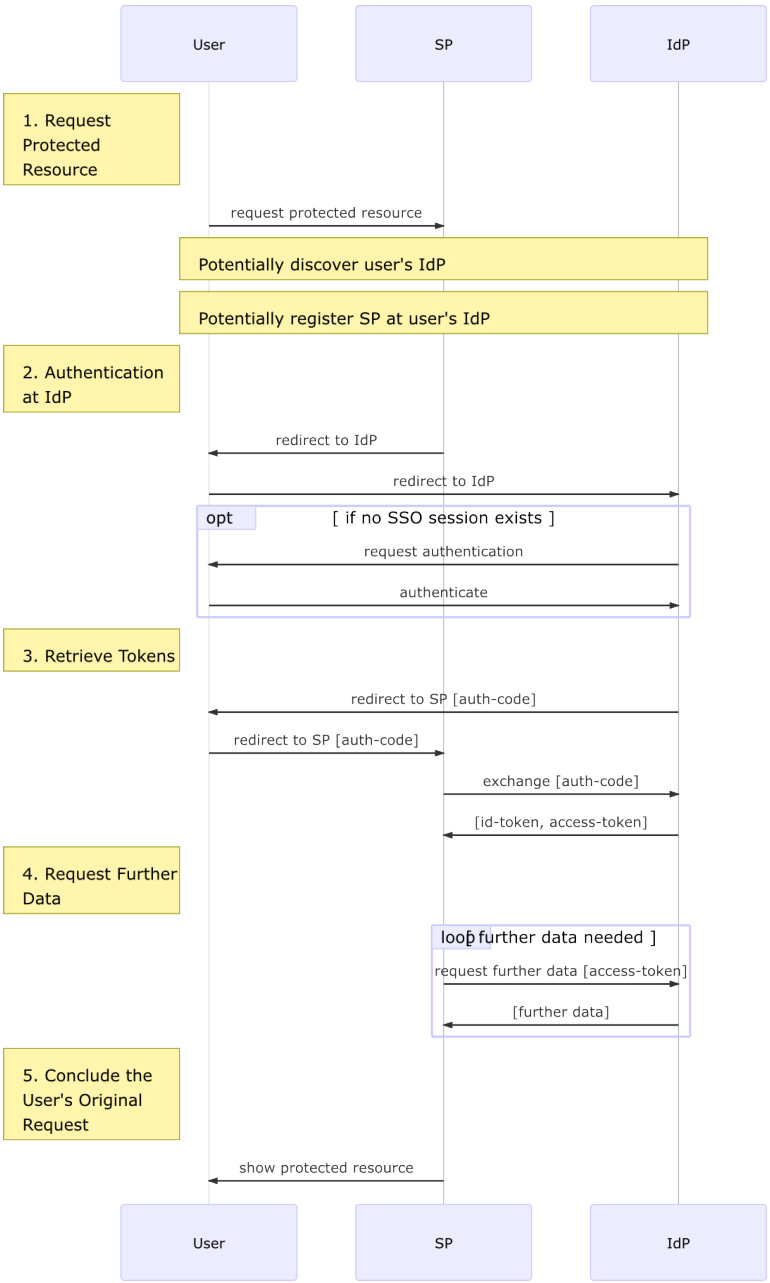


Figure 2.1.: This sequence diagram illustrates the individual steps during an authentication process with OpenID Connect. The whole process is started when a user tries to access a protected resource at the service provider. This service provider redirects the user to an identity provider. The association between service and identity provider has potentially been established through the discovery and dynamic registration extensions. Then, the user authenticates at the identity provider. An id-token representing the user's authentication and attributes, as well as an access-token demonstrating her authorization is issued. Subsequently, these tokens have to be retrieved by the service provider. With those tokens, the service provider is able to request further data and, ultimately, can respond to the user's initial request.

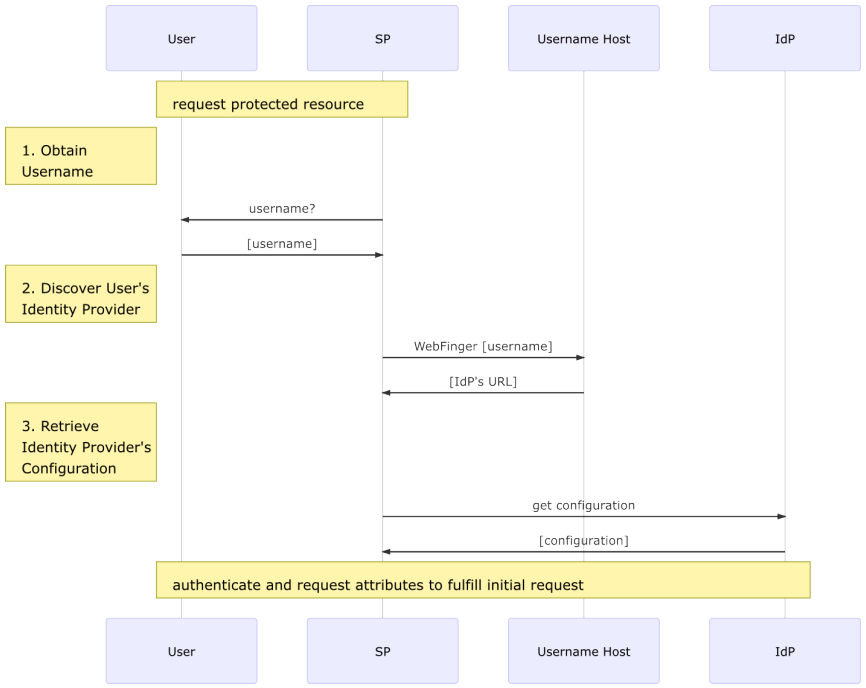


Figure 2.2.: This sequence diagram illustrates the discovery process of OpenID Connect Identity Providers. First, the service provider has to obtain the user's username, which contains the name of a host. Then, a WebFinger request to this host reveals the user's preferred identity provider. Finally, the service provider is able to retrieve the identity provider's configuration.

OAuth 2.0 [JCM15]. According to the OAuth 2.0 Bearer Token specification, the access-token can be included in the request to demonstrate authorization. Figure 2.1 illustrates this Bearer Token usage. In the JSON Web Token profile for OAuth 2.0, authorization can be demonstrated with JWT tokens, such as the id-token.

5. **Conclusion of the User's Request:** After the service provider's requirements have been fulfilled, the initial request by the user for a protected resource can be fulfilled. The authenticity of the user and her attributes have been presented by the obtained id-token. In addition, the service provider was able to request further services offered by the identity provider.

2.2.2. Discovery Process

The discovery extension of the OpenID Connect specification allows to determine the user's preferred identity provider and its configuration. This discovery process is based on usernames that reference hosts. By querying such a specified host, the service provider learns the URL of the user's preferred identity provider. With this URL, the service provider is able to obtain the identity provider's configuration. The following lines describe the individual steps of the discovery process, as illustrated in figure 2.2.

1. **Obtain Username:** In order to perform the authentication process triggered by the user's

attempt to access a protected resource, the service provider first has to obtain the username. Therefore, the service provider presents the user a form asking for her username. Valid usernames follow schemes that include a hostname, such as `joe@example.com:8080` or `https://example.com/joe`.

2. **Discover User's Identity Provider:** The service provider sends a *WebFinger* [Jon+13] request to the host extracted from the username in order to determine the user's preferred identity provider. This request is sent to a standardized endpoint and contains the username. Based on this information, the *WebFinger* service responds with a URL to the user's preferred identity provider.

3. **Retrieve Identity Provider's Configuration:** In the final step, the service provider retrieves the identity provider's configuration. The previously obtained URL specifies the identity provider's host. A request to a standardized path at this host returns configuration data, such as necessary endpoints, supported cryptographic algorithms, and the locations of key material.

2.2.3. Dynamic Registration Process

The OpenID Connect extension for dynamic registration allows to establish an association between service providers and identity providers. During this process the service provider presents information about itself to the identity provider and in turn obtains information required to use the identity provider. The exchanged information includes supported optional functionality as well as cryptographic algorithms and key material. As illustrated in figure 2.3, the extension for dynamic registration specifies two steps, which are described in further detail below.

1. **Register Service Provider at Identity Provider:** In order to register, the service provider sends information about its configuration to the identity provider. Additionally, based on the previously obtained identity provider's configuration, the service provider is able to select supported cryptographic algorithms. The identity provider takes these options into account, but ultimately defines which methods are used. As a result, the service provider receives a document specifying the parameters of the association, as well as a registration access token, which can be used to obtain the registration information at a later point in time.

2. **Read Registration Information:** With the previously obtained registration access token, the service provider is able to get a new copy of the registration information.

2.2.4. Session Management

Even though identity and service provider maintain their own sessions, an overarching concept of session management allows to coordinate these sessions. The identity provider might offer a long-term single sign-on session, which is opened when the user first authenticates, in order to skip the need for user interaction during immediate further requests. Service providers

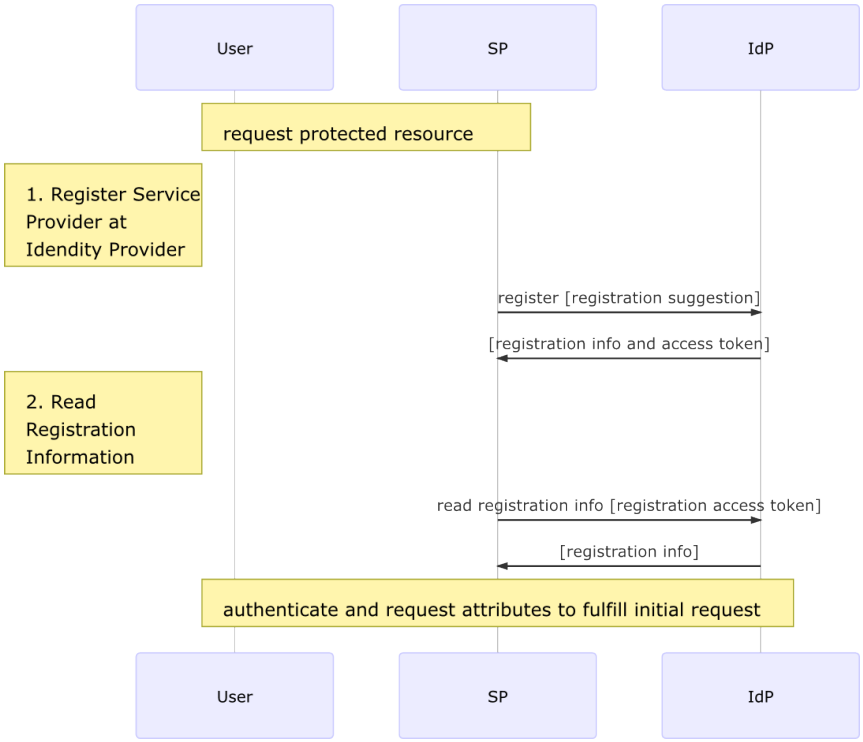


Figure 2.3.: This sequence diagram illustrates the communication defined for the dynamic registration extension of OpenID Connect. To register a new service provider at an identity provider, the parameters of their association have to be negotiated. The service provider suggests registration options, while the identity provider ultimately decides on the conditions. The response also includes a registration access token, which can subsequently be used to obtain a new copy of the registration information.

typically open a session after receiving and verifying the id-token issued by the identity provider. However, the service provider's session is derived from the identity provider's session, which exists because of the user's actual authentication. Therefore, identity and service provider have reason to coordinate their sessions.

The issues regarding session management and coordination were considered in further specifications for OpenID Connect. OpenID Connect Session Management [Med+16] defines how iframes can communicate through HTML5 Cross-Document Messaging [Hic15], in order to check the current session state and track changes. In addition, specifications for front-channel [Jon16] and back-channel [JB16] logout propose ways to close the identity and service provider's session simultaneously, indirectly via the user agent or, respectively, directly between identity and service provider.

2.3. Proxy Re-Encryption

Proxy re-encryption, introduced by Blaze, Bleumer, and Strauss [BBS98], enables a semi-trusted proxy to transform a ciphertext for one entity to a ciphertext for another entity without revealing the underlying message to the proxy. This section first describes the operations involved in a proxy re-encryption scheme. Subsequently, the trust relationship regarding the proxy is explained. Finally, this section provides two application examples that highlight new possibilities compared to traditional encryption schemes.

In contrast to traditional encryption schemes, proxy re-encryption introduces two new operations, namely re-encryption and the generation of a re-encryption key. Diagram 2.4 illustrates the relationships between those operations and their data dependencies. The operations of a proxy re-encryption scheme are described below:

KeyGen$() \rightarrow (sk_X, pk_X)$**:** This operation generates and returns a key pair (sk_X, pk_X), containing a private key sk_X and a public key pk_X.

ReKeyGen$(sk_A, sk_B \text{ or } pk_B) \rightarrow (rk_{A \rightarrow B})$**:** This operation takes the private key sk_A of Alice and key material of Bob to create a re-encryption key $rk_{A \rightarrow B}$ that is used to transform data encrypted for Alice to ciphertexts for Bob. Depending on the particular scheme, the key material provided by Bob has to be either his private or public key. The output of this operation is the re-encryption key $rk_{A \rightarrow B}$.

Enc$(M, pk_A) \rightarrow (C_A)$**:** Given a public key pk_A and a message M, this operation encrypts the message into a ciphertext C_A for the owner of the key pair to which pk_A belongs. The output of this operation is the ciphertext C_A.

ReEnc$(C_A, rk_{A \rightarrow B}) \rightarrow (C_B)$**:** The parameters of this operation are a re-encryption key $rk_{A \rightarrow B}$ and a ciphertext C_A for Alice, where the ciphertext was encrypted for the same user that provided her secret key in the re-encryption key generation. This operation transforms the ciphertext C_A into a ciphertext C_B for Bob, who presented the second key material in the re-encryption key generation. The output of this operation is the ciphertext C_B.

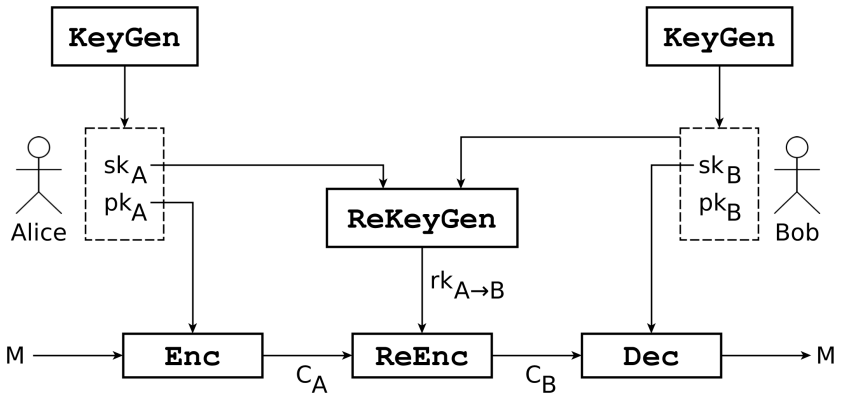


Figure 2.4.: This diagram illustrates the data dependencies, and thereby possible process steps, of a proxy re-encryption scheme. In the setup phase, both users, Alice and Bob, generate their key pairs and subsequently provide parts of that key material to generate a re-encryption key. Alice encrypts a message for herself. The resulting ciphertext can be re-encrypted for Bob, given the corresponding re-encryption key. In the end, Bob is able to decrypt the ciphertext to receive the original message.

Dec$(C_B, sk_B) \rightarrow (M)$**:** On input of a secret key sk_B and a ciphertext C_B, this operation decrypts the ciphertext C into its underlying plain message M if C_B was encrypted for the user who owns the key pair that includes sk_B. The output of this operation is the plain message M.

Proxy re-encryption leads to a system where the proxy only has to be semi-trusted and where the entity for which the message was originally encrypted stays in control of granting re-encryption rights. The proxy performs the re-encryption operation without seeing the underlying plaintext or having direct access to secret key material. Therefore, the proxy does not have to be fully trusted. The entity for which the message was originally encrypted stays in control, as it generates a re-encryption key that is used by the proxy during the re-encryption operation.

The applications of proxy re-encryption are manifold. In order to outline the capabilities introduced by proxy re-encryption, the following paragraphs describe two example applications.

Email Forwarding: In the email forwarding example, Alice wants to forward her encrypted emails to Bob for the duration of her vacation. When using a traditional encryption scheme, Alice would have to hand her private key to Bob in order for him to decrypt her mails. However, by making use of proxy re-encryption, Alice can generate a re-encryption key and hand this re-encryption key to the email server, which re-encrypts all of Alice's incoming email for Bob. As a result, Bob is able to decrypt Alice's emails with his own private key without requiring knowledge of Alice's key material.

File Storage and Sharing: In the file storage scenario, Alice wants to 1) store her encrypted files on a cloud-based file server, 2) retrieve and decrypt those files, and 3) share them with multiple other parties.

With traditional encryption schemes, Alice would have to store multiple encryptions of the same file for different recipients on the file server in order to fulfill all those requirements. One encryption would have to be for herself, as she needs to retrieve and decrypt it.

Furthermore, she would have to encrypt and upload another version of the file for each party with which she wants to share the file.

In contrast, proxy re-encryption simplifies the realization of such a use case considerably. With proxy re-encryption, Alice just uploads one ciphertext, which is the encryption of the file for herself. Retrieval and decryption of this ciphertext is easily possible. In order to share such an encrypted file with another party, Alice generates a re-encryption key, which the file server uses to re-encrypt Alice's encrypted file for the other party.

In conclusion, proxy re-encryption enables the receiver of a ciphertext to delegate decryption rights to another entity with the help of a semi-trusted proxy, which does not learn the underlying plaintext or any secret key material. In contrast to traditional encryption schemes, proxy re-encryption introduces two additional operations, namely re-encryption and the generation of re-encryption keys. The re-encryption operation can be conducted on a proxy that is semi-trusted. The generation of re-encryption keys is performed by the original recipient of the ciphertext, who thereby stays in control of granting re-encryption rights. Those characteristics make way for new applications that would not be possible with traditional encryption schemes.

2.4. Cryptographic Service Interoperability Layer (CrySIL)

The Cryptographic Service Interoperability Layer (CrySIL), proposed by Reimair et al. for the web [RTZ15] and mobile environment [Rei+15], specifies a protocol to access cryptographic operations and key material on remote devices. This section will first describe challenges modern applications face and then use these challenges as motivation for CrySIL. Subsequently, CrySIL nodes and their individual components are explained. Finally, an example application should illustrate the benefits of using multiple CrySIL nodes.

Current applications running in heterogeneous environments face major challenges in order to provide security features. As these applications perform cryptographic operations, they are also confronted with required key material, which has to be provisioned, securely stored, and shared between various devices. In addition, novel advanced cryptographic primitives and protocols have to be implemented for multiple different platforms to be used by such applications. Since the capabilities of those different platforms vary wildly, those challenges represent a very complex task.

Motivated by these challenges, CrySIL defines a protocol for accessing cryptographic operations and key material. Requests for cryptographic operations are given to a local CrySIL node, which fulfills the request by itself or in cooperation with remote nodes. For example, if a required key is not locally present or the requested cryptographic algorithm is not implemented, the request for a cryptographic operation can be forwarded to another CrySIL node that holds the key material or provides the required algorithm. CrySIL nodes can be embedded into existing applications through standardized APIs, such as PKCS#11, the W3C Web Cryptography API or the Java Cryptography Architecture (JCA). In consequence, these applications are able to immediately profit from newly accessible key material and cryptographic operations. An overview of the interaction between CrySIL nodes is given in figure 2.5.

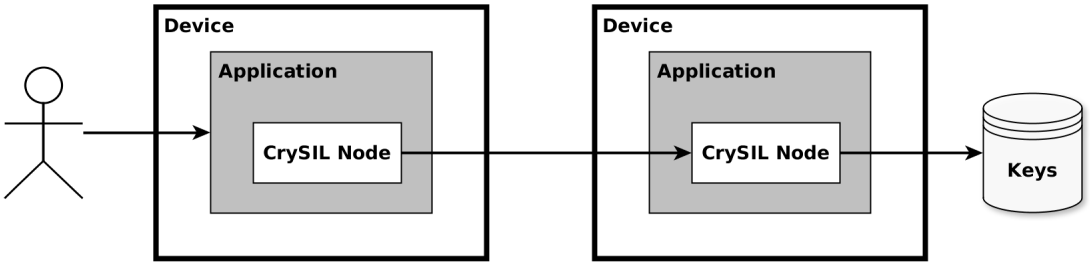


Figure 2.5.: This diagram illustrates the interaction between two CrySIL nodes. The user makes use of an application on her device. This application includes a CrySIL node for cryptographic operation. If this local node is not able to fulfill the cryptographic request, this request is forwarded to another CrySIL node that has the necessary means.

The CrySIL node represents the main building block of a distributed cryptographic system. Such a node provides cryptographic operations accessing key material, handles in- and outbound communication, and performs authentication and authorization. All these tasks are implemented by separate components in a modular design, which allows flexible extension. Figure 2.6 illustrates these modules and detailed descriptions are given below.

Receiver: *Receivers* accept cryptographic requests over different communication channels. Such a channel could be a cryptographic API, as defined by PKCS#11, the W3C Web Cryptography API or the Java Cryptography Architecture (JCA). This allows simple integration into application. Additionally, for the web environment, receivers could take requests via HTTP, HTML5 PostMessage or Web-Sockets. On the mobile phone, possible communication channels include push notifications or WebVPN, as described in [Rei+15]. After a request is received and parsed, it is forwarded to the router.

Router: The *Router* is a central component that determines which component is able to process the incoming request. This decision is typically based on the requested algorithm and key material. If the request can be processed locally, it is handed to an appropriate actor. Otherwise, the request is forwarded to another node with the necessary means.

Actor: *Actors* implement the offered cryptographic operations and have access to the involved key material. Basically, these actors adapt the incoming requests to function calls to cryptographic libraries supported by the respective platforms. Furthermore, the actors access the locally available key material, which is preferably stored in a secure location. These keys can either be protected by some platform specific mechanism, such as a Trusted Platform Module (TPM) on mobile phones, or they can be protected by the platform itself, as it is the case in an Hardware Security Module (HSM).

Communication: *Communication* modules forward requests that cannot be fulfilled locally to another CrySIL node that has the necessary means via a number of communication channels. For example, if the requested algorithm is not supported locally or the required key material is not available, the request is handed to another CrySIL node that implements the algorithm or holds the key material.

Authentication: *Authentication* is required, in order to determine whether a requester is au-

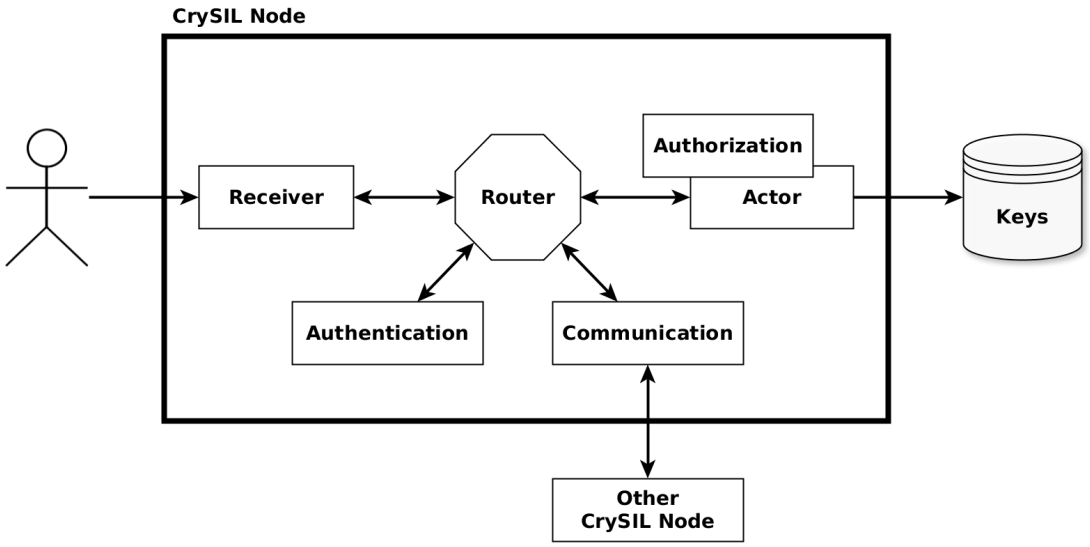


Figure 2.6.: This diagram depicts the individual modules of a CrySIL node. Receiver accept cryptographic requests from different communication channels. A router determines which component is able to process the request. Actors perform the actual cryptographic operations with access to key material. However, this access to key material also requires authentication and authorization. If the request cannot be fulfilled locally, the communication module sends it to another CrySIL node.

thorized to access the desired cryptographic operation and key material. During the processing of a client's request, the authorization engine may decide that the client has to provide initial or additional authentication information. In that case, the client is presented with an authentication challenge it has to stand by sending an appropriate response. For example, these challenges may require simple username and password pairs, tokens issued by external identity providers or multi-factor authentication responses.

Authorization: An *authorization* module protects access to the user's key material that is used in the offered cryptographic operations. Users specify authorization policies according to their requirements. With these policies, the authorization engine evaluates access control rights based on the client's authentication information, which also includes verified attributes. Therefore, if the engine requires further information about the client, it may trigger additional authentication challenges.

An example application for a CrySIL system would be a client-side web application that encrypts the user's files before uploading them to a cloud storage provider. In addition, this application downloads and decrypts the files in order to present them to the user. The javascript-based web client faces two challenges. Firstly, client-side storage of key material requires access to the platform features for a secure implementation and does not allow the user to use multiple devices. Secondly, if the user's files are encrypted in a complex format, such as the Cryptographic Message Syntax (CMS), there might not be an implementation in javascript. These challenges could be overcome by using one CrySIL node at the web client (web-node) and providing another node in a user-trusted environment (user-node), for example in an HSM. In such a setup, the web-node cooperates with the user-node by forwarding cryptographic requests it is

not able to fulfill. Firstly, by storing the user's key material at the user-node, the keys are stored securely, as the user-node employs its platform-specific protection mechanisms. In addition, the key material is decoupled from the device that is executing the client application and, therefore, can be used by other devices as well. If the user's files are encrypted with a simple hybrid encryption scheme, this can be implemented efficiently. The web-node generates a symmetric key which is used to encrypt the file's content. This content encryption key is then encrypted with the user's public key obtained from the user-node. In order to decrypt such a file, this whole process is reversed, but the encrypted content encryption key is sent to the user-node for decryption. Secondly, if the client uses a complex encryption scheme that is not supported, the web-node could outsource the whole encryption process to the user-node. As a result, CrySIL enables such a web application to encrypt and decrypt files, even without having the necessary key material or supporting the required encryption algorithms.

In conclusion, CrySIL nodes cooperate through a standardized protocol, in order to fulfill requests for cryptographic operations that access key material. These nodes consist of separate components that handle incoming and outgoing communication, provide cryptographic implementations which access key material, and perform authentication as well as authorization. By building a network of nodes, the platform-specific capabilities and trust assumptions in the deployment environment can be used to satisfy the application's requirements.

2.5. Chapter Conclusion

In this chapter, multiple technologies have been described that make up the basis of the proposed identity broker system. First, JSON Web Tokens and their use in the OpenID Connect protocol have been described. Then, the cryptographic primitive proxy re-encryption has been explained. Finally, the cryptographic service interoperability layer has been examined.

JSON Web Tokens (JWT), which depend on the JOSE standards, are a web-friendly format for encrypted and signed content. In the OpenID Connect identity protocol, a service provider delegates the authentication of users and the request for attributes to an identity provider. This protocol uses JWT to attest a successful result of the authentication process.

Proxy re-encryption extends public key encryption by enabling a proxy to transform a ciphertext for one entity to a ciphertext for another entity, without revealing the underlying plaintext in the process. This process requires consent from the original recipient of the ciphertext, as the recipient has to generate a re-encryption key from her private key. Proxy re-encryption has a multitude of applications, as it allows to delegate decryption rights.

Finally, a cryptographic service interoperability layer enables multiple nodes to cooperate through a standardized protocol, in order to fulfill cryptographic requests. In this approach, if a node is not able to fulfill a request for a cryptographic operation locally, it forwards this request to another node that possesses the necessary means, such as required key material or the implementation of the requested algorithm.

3. The Proposed Identity Broker System

The proposed identity broker system employs innovative solutions to achieve privacy and usability objectives. In particular, the user's privacy in the cloud is protected by re-encryption and user-friendly key management is realized by embedding a cryptographic service via CrySIL. This chapter presents the proposed identity broker system by explaining the objectives, the process flow, and the involved components.

The organization of this chapter is as follows: In the beginning, section 3.1 explains the objectives of the proposed identity broker system. The remainder of this thesis uses these objectives as basis for decisions and argue how the objectives are met. Then, section 3.2 describes the typical steps of an authentication and attribute sharing process. In section 3.3, the user experience during this process is examined. Subsequently, section 3.4 describes the involved components in further detail. Finally, section 3.5 concludes this chapter.

3.1. Objectives

The goal of this thesis is the design and development of a usable and privacy-preserving identity management system that can be operated in the cloud. Such an identity management system has fulfill functional requirements, namely to perform authentication and provide access to the user's attributes. However, the aspiration for privacy and usability leads to additional objectives. Those objectives are used in the remainder of this thesis as a basis for decisions.

Privacy is a major concern when handling sensitive identity data, especially in the cloud environment. From the need for privacy, we can derive the following objectives:

O.1 **Confidentiality:** As a cloud operator has the power to inspect all data that are handled in the cloud, the user's sensitive attributes have to be encrypted in order to ensure their confidentiality. This includes attributes that are stored at an attribute provider as well as attributes during the sharing process.

O.2 **Explicit Consent:** In order to implement a user-centric system, the user has to provide explicit consent to enable the sharing of her attributes.

O.3 **Minimal Data Disclosure:** To protect the user's privacy, only data that is required by the service provider to fulfill its task should be disclosed.

However, in order to achieve user-adoption, the identity management system not only needs to be privacy-preserving but also easy to use. This usability requirement leads to additional objectives.

O.4 **Minimal Memorization:** A prerequisite for a comfortable user experience is that user only have to memorize a minimal set of data. Such data include credentials of which the user demonstrates knowledge during authentication.

O.5 **Minimal Interactions:** Users want to complete processes as fast as possible with a minimum number of interactions. This objective especially concerns repeatedly entering the same data or performing the same process steps.

O.6 **User-Friendly Key Management:** As cryptographic operations are involved in the proposed solution, the user has to store and use key material. Key management represents a major obstacle, as the involved concepts and security implications are most likely unfamiliar to users. Therefore, an objective of this thesis is to realize key management in a user-friendly way.

To summarize, there are several objectives to be considered in the design of a usable and privacy-preserving identity management system. To preserve the user's privacy, the confidentiality of the user's attributes has to be ensured, while explicit consent is needed to share a minimal required set of these attributes. However, usability is an important aspect for the adoption by users. Therefore, the user should only be required to remember a minimal amount of data and the number of interactions should also be minimized. In addition, the involved storage and access to the user's keys requires a user-friendly solution.

3.2. Process Steps

The complete process of authentication and data sharing in the proposed identity broker system is illustrated in figure 3.1. Some of the involved steps can be skipped due to usability improvements, as explained in further detail in the next section 3.3. The following lines describe the individual process steps.

0. Initially, the system has to be set up. Parts of this step only have to be performed if a new participant joins the identity broker system. In order to use proxy re-encryption, service providers and users require key pairs. Service providers create their key material locally, whereas users generate and store the key pairs at their CrySIL nodes. Users encrypt their attributes before uploading them to an attribute provider. This encryption is either preformed by the CrySIL node that holds the user's key material, or the user retrieves her public key and uses it to encrypt at another location.

1. To initiate the actual process, the user attempts to access a protected resource on the service provider. In order to fulfill the user's request, the service provider requires users to authenticate and to share some of their attributes. Therefore, the service provider delegates the authentication and data acquisition by forwarding the user to her broker.

2. The broker forwards the user to her preferred identity provider to perform an authentication process. Her decision on which identity provider should be used, as well as subsequent selections are stored in a user profile. Those profiles enhance usability by minimizing the required user interaction.

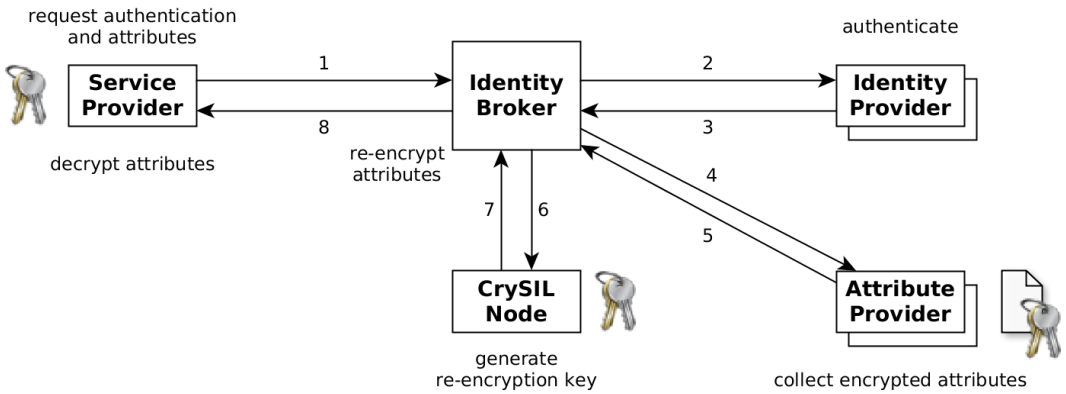


Figure 3.1.: This diagram illustrates the architecture and general process flow of the proposed identity broker system. The service provider requires authentication and user's attributes to a service to the user. (1) Therefore, the user is forwarded to the broker. (2,3) The broker delegates authentication to an identity provider and (4,5) collects the user's encrypted attributes from attribute providers. (6,7) A re-encryption key is generated via a CrySIL node, in order to re-encrypt the encrypted attributes. (8) These re-encrypted attributes can subsequently be decrypted by the service provider. Finally, the service provider is able to fulfill the user's request.

3. The authentication process on the identity provider yields a result, which is sent back to the broker.

4. The broker requests the required attributes from the user's attribute providers.However, the broker needs authorization to access these attributes. Therefore, the user might have to authenticate at the attribute provider and grant her consent.

5. The attribute providers respond with the requested attributes. These attributes have been encrypted for the user with her private key before depositing them at the attribute provider.

6. A re-encryption key is needed to share the encrypted attributes with the service provider. Therefore, the broker asks the user-specified CrySIL node to generate such a re-encryption key that transforms ciphertexts for the user into ciphertexts for the service provider. In addition, the result of the authentication process and the service provider's public key are also sent to the CrySIL node.

7. The CrySIL node evaluates its policies against the data sent in the request. Those policies state that the authentication information has to match the involved public key and that the provided public key belongs to a valid service provider. If those conditions are satisfied, the stored private key of the user and the received public key of the service provider are used to generate a new re-encryption key. The resulting re-encryption key is sent back to the broker.

8. The broker re-encrypts all encrypted attributes and sends them along with authentication information back to the service provider, which is able to decrypt the attributes. Finally, the service provider is able to fulfill the user's initial request.

3.3. User Experience

In order to authenticate and to share attributes, the user has to interact with components at multiple points. By storing the outcomes of those interactions, they can be skipped in subsequent runs. The following lines describe the individual procedures that require the user's attention. After that, typical scenarios are examined regarding their required interaction.

Consent at the Broker: If a service provider wants a user to authenticate and share data, the user's consent is needed. The user is informed by the broker about which data are requested by the service provider. Depending on the user's preferences this request for consent can be omitted in subsequent runs.

Build Profile at the Broker: The user has to select which identity provider should be used as well as how her attributes should be retrieved. Those decisions are stored in a user profile and, therefore, can be omitted in following runs.

Authenticate at the Identity Provider: In order to authenticate, the user has to interact with the identity provider to prove one or more authentication factors, which include knowledge, possession or inherence. This authentication is valid for a short timespan during which further authentication requests can be fulfilled through a single sign-on session that is bound to the user. Therefore, subsequent authentication requests in the validity period do not require user interaction.

Authorize at the Attribute Provider: The user has to advice the attribute provider to allow the broker to access her attributes. For this authorization procedure, the user usually has to be authenticated at the attribute provider as well. In the end, the broker receives an access token, which represents the authorization granted by the user for a certain timespan. In some scenarios, the broker obtains authorization without user interaction. If the access token is still valid, the user neither has to authenticate or authorize. Otherwise, the broker has to obtain a new access token. For authentication, an existing single sign-on session from either the attribute provider or a trusted identity provider might be reused. Depending on the user's security policy, a previous authorization decision can be used. As a result, access to the attribute provider requires the user's authentication and authorization, but these interactions can be omitted in some scenarios.

Generate Re-Encryption Key at the CrySIL Node: In order to allow sharing of encrypted attributes, the user has to generate a re-encryption key on her CrySIL node. Depending on the user's security policy as well as the chosen deployment environment of the CrySIL node, some user interaction might be necessary. For example, if the CrySIL node is deployed on the user's phone, a click on a button might be enough to consent to generating a re-encryption key from the user's private key. A CrySIL node in the cloud might require consent through a web form. Otherwise, the user could grant key generation rights if the broker is able to provide the user's authentication data. Therefore, there are multiple scenarios, where re-encryption key generation can be achieved without user interaction. Nonetheless, this key generation only has to happen one time per service provider, as the re-encryption key is stored together with the user's profile.

In the following paragraphs, we examine the required user interaction in three typical scenarios. The performed steps depend on the availability of user profiles, single sign on session, and re-encryption keys, as well as, the user's preferences. Table 3.1 summarizes the required user interactions for the typical scenarios.

First Authentication: During the first time a user uses the identity broker to provide authentication and attributes to a service provider, the most interaction is required. The broker has not yet compiled a user profile and the user did not generate a re-encryption key for this service provider thus far. Further, let us assume the user does not have any single sign on sessions. Thus, the user has to give consent at the broker, make decisions to build a profile, authenticate at the identity provider and grant access to the attribute provider. Depending on the user's security policy and chosen deployment environment for the CrySIL node, some interaction in the re-encryption key generation might also be necessary.

Subsequent Authentication with Single Sign On Sessions: If the user recently provided authentication information and attributes to the service provider via the broker, most procedures requiring interaction can be skipped. As the user has been using the service provider before, the broker has already acquired consent, compiled a profile and stored a re-encryption key. Let us further assume, that the last run has been performed recently, and single sign on sessions for identity provider and access token for attribute provider are still valid. As a consequence, the user neither has to interact with the identity nor the attribute provider. Depending on the user's preferences, only the broker would ask for the user's consent to notify her about this process.

Subsequent Authentication without Single Sign On Sessions: Under some circumstances, the user might not be able to reuse some authentication and authorization information in a subsequent run and, therefore, additional interaction might be required. For example, the user's single sign on sessions and access tokens expire after some time. Further, if the user makes use of another device, previously opened single sign on sessions are not available. In such a case, it is necessary to re-authenticate at the identity provider and to grant access to the attribute provider again. However, the broker still has the user profile and re-encryption key and, therefore, these steps can be skipped. Again, depending on the user's preferences, the broker might ask for the user's consent.

In conclusion, to authenticate and share attributes via the proposed identity broker system, the user has to interact with the system's components in multiple steps. These steps include giving consent at the broker, selecting identity and attribute providers, authenticating, authorizing access to the attributes, and, finally, generating a re-encryption key. However, after the initial use of the proposed system with a service provider, the number of interaction steps can be significantly reduced. In the best case, only optional consent has to be given at the broker. Infrequently, the user has to re-authenticate at the identity provider and grant access to the attribute provider anew.

	Consent at Broker	Build Profile at Broker	Authenticate at Identity Provider	Authorize at Attribute Provider	Key-Generation at CrySIL node
First Authentication	✓	✓	✓	✓	?
Subsequent Authentication with SSO Sessions	?				
Subsequent Authentication without SSO Sessions	?		✓	✓	

Table 3.1.: This table illustrates the required user interactions in different scenarios. During the first time a user makes use of a service provider, multiple steps require user interaction. In subsequent procedures, steps can be skipped due to compiled profiles and existing single sign on session. Some interaction might be required depending on the user's preferences (denoted with a questionmark).

3.4. Components

This section provides a high-level description of the major components that make up the proposed identity broker system. Section 3.4.1 describes the service provider, which requires authentication and attributes, and thus forwards the user to the identity broker. This identity broker, explained in section 3.4.2, interacts with three other components to perform authentication, acquire attributes and generate a re-encryption key. Firstly, the broker delegates authentication to identity providers, which are described in section 3.4.3. Secondly, attributes are obtained from attribute providers, which are explained in section 3.4.4. Finally, a CrySIL node, presented in section 3.4.5, generates re-encryption keys for the broker.

3.4.1. Service Provider

In order to offer its service, the service provider requires users to authenticate and to share data. For example, an online copyshop has to be able 1) to authenticate the user to link her with her orders, and 2) to access the user's data such as the documents to print, the delivery address or billing information. As described in the following paragraphs, the service provider enjoys various benefits by outsourcing those tasks to an identity provider.

The service provider forwards users to a trusted identity broker for authentication. Authentication processes are hard to implement securely and are a prominent target for attackers because of the involved sensitive data. By delegating authentication to the identity broker, the service provider also hands off these security challenges. In addition, using an identity broker improves usability. If the broker supports services where the user is registered, the user can reuse an existing account and, therefore, does not have to remember a new set of credentials for the service provider. Further, if the user is already logged into that existing account, she

does not have to enter her credentials again, which reduces the required interaction. As a result, delegating the authentication process has security and usability benefits.

Additionally, the service provider gains access to the user's shared data via the identity broker. With the user's consent, the broker consolidates encrypted attributes from various sources. Those encrypted attributes are re-encrypted with the result that the service provider is able to decrypt the required data. Consequently, the service provider gets access to data, while the user stays in control of the sharing process and enjoys usability benefits, as she does not repetitively need to re-enter the same data.

In conclusion, the service provider outsources authentication and data acquisition to an identity broker. By doing so, the service provider experiences multiple positive effects, especially regarding usability and security.

3.4.2. Identity Broker

The cloud-based identity broker offers user authentication and access to the user's data to service providers, while the user stays in control of the data sharing process. In the following paragraphs, 1) the authentication and data sharing is described, 2) the user's control mechanisms are explained, and 3) the consequences of operating an identity broker in the cloud are explored.

Authentication and access to the user's attributes is provided by the identity broker. Service providers forward users to the broker with a request for authentication and access permissions. If no single sign on session is available, the user is sent to a supported identity provider for authentication. Once this authentication succeeded, the broker retrieves encrypted attributes from various attribute providers. Subsequently, those encrypted attributes are re-encrypted for the requesting service provider. The broker returns the user's browser to the service provider, sends authentication information and offers access to the re-encrypted attributes.

Nevertheless, the user stays in control of this authentication and sharing process by defining a policy enforced by the broker and through the regulated generation of proxy re-encryption keys. The user is able to specify a policy for each service provider. For authentication, the user selects one of the supported identity providers. In addition, consent for the requested permissions has to be obtained and the user is able to specify from which attribute provider the encrypted data should be retrieved. In order to minimize the required interaction, all those decisions are compiled into a profile the first time a service provider forwards the user to the broker. Furthermore, with cryptography, the user literally holds the key to enable data sharing. That is, the user wields her private key to generate a re-encryption key. This private key is stored in a CrySIL node, which exposes the re-encryption key generation process. Only after obtaining this re-encryption key, the broker is able to re-encrypt the user's data for one specific service provider. As a result, with policies and cryptography the user has control over authentication and data sharing.

Operating such an identity broker in the cloud offers multiple benefits, but also has an impact on the trust relationships. The use of cloud infrastructure provides many advantages, such as scalability, high performance and availability, as well as a pay-as-you-go pricing model to name a few. In consequence, the cloud is an attractive deployment target. However, cloud

computing comes at a cost. As the cloud infrastructure is operated by another provider, services running in this environment become semi-trusted. An identity broker in the cloud can be considered honest-but-curious, since the cloud operator might inspect the data it processes. As a consequence, the service provider trusts the broker to be honest and therefore to correctly perform authentication via an identity provider. However, the user suspects the broker to be curious and therefore protects her attributes with a re-encryption scheme. As a result, the benefits of operating an identity broker in the cloud can be utilized if re-encryption is used to overcome trust issues.

In conclusion, a cloud-based identity broker offers user authentication and access to the user's data to service providers. To achieve this, the broker integrates identity providers and attribute providers. The user stays in control by defining policies and by using cryptography. Also, the benefits of cloud computing can be utilized, as re-encryption protects the user's attributes in a semi-trusted environment while still allowing to share those attributes.

3.4.3. Identity Providers

Identity providers perform the authentication process for other parties. In comparison to implementing authentication directly at a service provider, an identity provider can offer usability benefits. Further, with an identity provider, security, policy and legal aspects can be satisfied. In the proposed identity broker system, the broker delegates the user authentication to such identity providers.

In general, an identity provider authenticates users for another party. In this authentication process, the user has to prove one or more authentication factors, which include knowledge, possession or inherence. After successful authentication, the identity provider usually issues some kind of token, which can be used to verify the authentication.

Identity providers offer usability benefits over implementing authentication processes directly into the services. Single sign-on allows to reuse the authentication information on multiple services as soon as a user has signed in once at an identity provider. Therefore, the user does not have to remember multiple credentials, one for each service, but only one for the identity provider. However, in order to sign in, the user requires an account at the identity provider. Many widely used internet services, such as Google [Goo15b] and Facebook [Fac16], offer the functionality of an identity provider. By integrating such identity providers into the broker, a significant percentage of possible users can be covered.

In addition, using identity providers can yield various security features and, thereby, satisfy the requirements of specific applications. One such security feature would be multi-factor authentication, where a user proves multiple authentication factors, which results in a strong authentication. Further, when a corporation-operated identity provider is used, the corporation can specify policies that are tailored to their use cases. Finally, by integrating a national identity solution, the authentication can have legal implications.

In the proposed identity broker system, the broker delegates the user authentication process to identity providers it trusts and by doing so enjoys usability and security benefits. As various types of identity providers can be integrated into the broker, the user is presented with a

usable solution that offers single sign on to a significant part of possible users. Additionally, the integration of company identity providers or national identity solutions enables use cases that demand specialized security requirements.

3.4.4. Attribute Providers

Attribute providers store the user's data and grant access to portions of the attributes after obtaining the user's consent. The following two paragraphs 1) describe the general purpose of attribute providers including security and usability aspects, and 2) explain ways to minimize the required user interaction.

In general, attribute providers store the user's data and share it with entitled entities. In the proposed identity broker system, only encrypted data is stored at the attribute provider, which makes the user's data safe from prying eyes. In addition, storing data at an attribute provider has the benefit that those attributes are available from anywhere and do not have to be on the current device.

However, to share the user's attributes, the user has to authenticate and, subsequently, grant sharing permissions. If the attribute provider delegates the authentication process to an identity provider, single sign on allows to reuse existing authentication information. In consequence, this authentication process might not require user interaction. The user only has to grant consent to share data, but depending on the particular implementation, this decision might also be stored, resulting in no further interaction.

In conclusion, the user's data can securely and conveniently be stored on an attribute provider. Such an attribute provider shares the user's data with entitled entities. In order to enable this sharing, an authenticated user has to grant access. Depending on the implementation of the attribute provider, the required user interaction can be minimized.

3.4.5. CrySIL Node

The cryptographic service interoperability layer (CrySIL) connects nodes to satisfy the cryptographic needs of the proposed identity broker system. The following paragraphs 1) briefly summarize the capabilities of CrySIL nodes 2) explain the purpose of a CrySIL node in the proposed identity broker system 3) describe the implications of the trust requirements on the execution environment, and 4) illustrate the involved authorization process.

In general, a CrySIL node performs cryptographic operations by itself or by cooperating with other CrySIL nodes. First, a node tries to fulfill the request with the algorithms and key material locally available. However, if the request cannot be fulfilled locally, it is forwarded to another node that supports the requested operation or has access to the required key material.

In the identity broker system, the broker asks the CrySIL node to generate a re-encryption key whenever the user makes use of a service provider for the first time. This operation involves the service provider's public key and the user's private key. The resulting re-encryption key allows the broker to re-encrypt the user's encrypted attribute for the service provider.

As the user's private key is required in the re-encryption key generation, this operation has to be performed in a fully trusted environment. Two possible options would be to deploy the CrySIL node in a trusted cloud environment or to execute it on the user's phone. In the cloud scenario, the user would have to place trust in the cloud operator to employ appropriate security mechanism such as the use of a hardware security module. A CrySIL node on the user's phone would give complete control to the user.

In order to use key material in the requested operations, authentication and authorization are required. As mentioned above, for the generation of a re-encryption key, the user's private key is required. Therefore, the user is able to define a policy with requirements controlling the access to her private key. Those policies depend on the user's trust in the requester. In the proposed identity broker system, the broker has to present valid authentication information obtained from an identity provider.

In conclusion, the CrySIL node generates re-encryption keys whenever a user makes use of a service provider for the first time. As the user's private key is involved in the generation, the node has to be operated in a fully trusted environment. Further, authorization to access the key material is governed by policies.

3.5. Chapter Conclusion

In this chapter, a privacy-preserving and usable identity broker system has been proposed. Privacy and usability lead to additional objectives, which have served as basis for the design process. Those objectives have been taken into account in the interactions between multiple components.

In order to realize a usable system, service providers delegate the authentication and attribute acquisition process to the identity broker, which performs authentication with an identity provider and retrieves the required attributes from attribute providers. In consequence, the amount of information the user has to memorize is reduced, as the user only has to remember one set of credentials for her preferred identity provider. Additionally, the user does not have to interact with service providers to repeatedly enter data, such as addresses or payment informations, but can just share those attributes. Even though the proposed system introduces additional interactions, most of them can be skipped in typical use cases. Further, usable key management is achieved by integrating a CrySIL node that is under the control of the user. Such a CrySIL node can either be a trusted cloud service or an app on the user's phone. As a result, the proposed system should be usable.

Privacy is mainly accomplished through the use of proxy re-encryption. The confidentiality of attributes is ensured, as encrypted attributes are uploaded to the attribute provider and they are shared by re-encrypting them for a service provider, which does not reveal the underlying plaintext at any intermediate point. In order to enable this re-encryption, the user has to generate a re-encryption key which represents the user granting explicit consent. In addition, a minimal amount of data can be disclosed by separately encrypting and storing each attribute at the attribute provider. In consequence, the broker only retrieves, re-encrypts and, thereby, discloses

the minimal set of attributes to the service provider. As a result, the proposed system protects the user's privacy.

4. Evaluation of Proxy Re-Encryption Schemes

This chapter evaluates multiple proxy re-encryption schemes based on property requirements. Proxy re-encryption is an integral cryptographic technology for the proposed identity broker system. This cryptographic principle allows to share the user's encrypted data by enabling a semi-trusted broker to re-encrypt that data for an authorized service provider without being able to read the underlying plain data in an intermediate step. However, there are multiple proxy re-encryption schemes, which have different properties. Therefore, requirements regarding these properties are identified in order to subsequently select suitable proxy re-encryption schemes.

This chapter is organized as follows: Section 4.1 provides an informal explanation of the most important properties of proxy re-encryption schemes. The following section 4.2 describes multiple re-encryption schemes and assesses their properties. In section 4.3, the requirements regarding properties of proxy re-encryption schemes for the use in the proposed identity broker system are identified. Section 4.4 evaluates the described proxy re-encryption schemes based on the identified property requirements. Finally, this chapter is concluded in section 4.5.

4.1. Properties

This section provides an informal explanation of the most important properties of proxy re-encryption schemes. These properties are later the basis for the evaluation of different proxy re-encryption constructions.

Unidirectional, bidirectional: A proxy re-encryption scheme is *unidirectional* [ID03] if a delegation of decryption rights from Alice to Bob does not also allow re-encryption from Bob to Alice. If such a delegation also enables re-encryption in the opposite direction, then the scheme is *bidirectional*.

Single-hop, multi-hop: A proxy re-encryption scheme is *single-hop* [CH07] if an initial ciphertext can only be transformed into a re-encrypted ciphertext, but further re-encryption of that already re-encrypted ciphertext is not possible. In contrast, a proxy re-encryption algorithm is *multi-hop* if an already re-encrypted ciphertext can be further re-encrypted.

Collusion-safe (Master Key Security): A proxy re-encryption scheme is *collusion-safe* [Ate+06] if, even though the proxy and a re-encryption recipient collude by sharing their key material, they are not able to learn the sender's private key.

Non-interactive: A proxy re-encryption scheme is *non-interactive* [Ate+06] if the delegator is able to generate a re-encryption key without having to interact with the delegatee or a trusted third party. Since the delegatee does not want to expose her private key, the

delegatee's private key material cannot be involved in the re-encryption key generation of a non-interactive proxy re-encryption scheme.

Key-private: A proxy re-encryption scheme is *key-private* [ABH08] if an adversary is not able to identity any participant from a re-encryption key, even when given all public keys and extensive interaction abilities. These abilities include being able to obtain the re-encryption of any chosen ciphertext, as well as the ability to obtain any re-encryption key except for the re-encryption key being analyzed.

Identity-based: Identity-based cryptography is a sub-category of public-key cryptography in which the public key of an entity can be derived from identity information about that entity, such as an email address. An identity-based system usually relies on a trusted third party which issues private keys to authorized entities. These private keys are derived from a master key and identity information. Therefore, a proxy re-encryption scheme is considered to be *identity-based* [GA07] if publicly known identity information represents an entity's public key.

Pairing-based, Lattice-based: Proxy re-encryption schemes rely on mathematical principles, which have fundamental security implications regarding the scheme's resilience against cryptographic analysis. The following two paragraphs describe lattice- and pairing-based algorithms.

Lattice-based [XT10] algorithms rely on lattices as underlying mathematical principle. Lattices are of particular interest, since no algorithm has yet been discovered that could harness the benefits of quantum computers in the cryptographic analysis of lattices.

Pairing-based algorithms use pairings as mathematical foundation. In contrast to lattice-based algorithms, cryptographic analysis on pairing-based algorithms could be performed efficiently on a quantum computer. One of those algorithms for cryptographic analysis is described by Shor [Sho97].

CPA-secure: An encryption scheme is *CPA-secure* [Ate+06] if an adversary is not able to match one of two plaintexts to an observed ciphertext, even when given the power to perform chosen-plaintext attacks (CPA). The next paragraph, which describes a chosen-plaintext attack, is followed by a more detailed definition of the CPA-secure property via the CPA game.

In a chosen-plaintext attack, the adversary is able to choose any plaintext, which is then encrypted. The resulting ciphertext can be observed by the adversary. As a consequence, the adversary has access to an encryption oracle. This strong definition captures all cases in which an adversary is able to influence the plaintext and observe any influence on the corresponding ciphertext.

The CPA-secure property can be described with the CPA game. In the three steps of this game, the adversary is given access to an encryption oracle. Firstly, the adversary defines two messages of same length. Secondly, the challenger randomly chooses one of those messages and encrypts it. Thirdly, the adversary observes the ciphertext and has to guess which of these messages has been randomly chosen for encryption. The adversary wins the game if the guess was correct. An encryption scheme is *CPA-secure* if an adversary is

not able to win the CPA game with non-negligible probability.

CCA-secure: An encryption scheme is *CCA-secure* [LV08; CH07] if an adversary can perform chosen-plaintext as well as chosen-ciphertext attacks (CCA) but is still not able to match one of two plaintexts to an observed ciphertext. Since the adversary is also given the ability to perform chosen-plaintext attacks, the CCA-secure property is stronger than the CPA-secure property. Therefore, any scheme that is CCA-secure is also CPA-secure. The next paragraph describes a chosen-ciphertext attack. Subsequently, a more detailed definition of the CCA-secure property via the CCA game is given.

In a chosen-ciphertext attack, the adversary is able to choose any ciphertext and then observe the corresponding decrypted plaintext. Therefore, the adversary has access to a decryption oracle. This gives extensive powers to an attacker, which encompass all more practical attacks where an adversary only has limited abilities to influence a ciphertext and to observe its influence on the decryption.

The CCA-secure property can be described with the CCA game, which is very similar to the CPA game explained above. However, the adversary is not only given access to an encryption oracle but also has access to a decryption oracle during the whole process. The only limitation regarding the decryption oracle is that the challenge ciphertext can not be decrypted. The three steps of this game are essentially the same as stated in the CPA game. Firstly, the adversary defines two messages of same length. Secondly, the challenger randomly chooses one of those messages and encrypts it. Thirdly, the adversary observes the ciphertext and has to guess which of the message has been chosen randomly for encryption. The adversary wins the game if the guess was correct. An encryption scheme is *CCA-secure* if an adversary is not able to win the CCA game with non-negligible probability.

In conclusion, this section provided an informal description of the most important properties of proxy re-encryption schemes. Those properties are used in the next section to assess multiple proxy re-encryption schemes.

4.2. Proxy Re-Encryption Schemes

This section describes multiple re-encryption schemes and evaluates their properties, which have been explained in the previous section 4.1. These schemes are listed in a roughly chronological order to outline the evolution of the research field of proxy re-encryption. The first constructions are mostly pairing-based, whereas later schemes are identity-based and finally lattice-based. In conclusion, the properties of the individual schemes are summarized in table 4.1.

In 1998, Blaze, Bleumer, and Strauss [BBS98] introduced the concept of proxy re-encryption. In their elgamal-based scheme, a re-encryption key can be used in both directions, making it *bidirectional*. It allows to further re-encrypt already re-encrypted ciphertexts. Therefore, this scheme is *multi-hop*. However, this first scheme has multiple shortcomings. It is a) *not non-interactive*, since the recipient's secret key is required in the re-encryption key generation, and b) *not collusion-safe*, since the sender's secret key can be recovered from the re-encryption key if the

proxy colludes with the recipient. Furthermore, their construction is only *CPA-secure* and, since it is deterministic, it is *not key-private*.

Ateniese et al. [Ate+06] proposed a new scheme based on bilinear pairings, which is *unidirectional* and *single-hop*. They succeeded in solving the major problems of the scheme by [BBS98]. Their construction is *non-interactive* as well as *collusion-safe*. However, this scheme is still only *CPA-secure* and, due to its determinism, *not key-private*.

The first *CCA-secure* proxy re-encryption scheme was proposed by Canetti and Hohenberger [CH07]. Their construction is *bidirectional* and *multi-hop*. However, it suffers the same major problems as the scheme by [BBS98]. That is, this scheme is *not collusion-save* and *not non-interactive*. In addition, it is *not key-private*.

Libert and Vergnaud [LV08] presented another *CCA-secure* proxy re-encryption scheme. Their construction is *unidirectional* and *single-hop* as well as *collusion-save* and *non-interactive*. However, it is *not key-private* even though the re-encryption operations are randomized via a blinding factor.

Hohenberger et al. [Hoh+07] proposed to use obfuscation to realize *unidirectional*, *single-hop* proxy re-encryption. In their approach, they define re-encryption as a program that simply decrypts the input ciphertext with the sender's private key and then encrypts the plaintext with the recipient's public key. Such a program using hardcoded key material is obfuscated. That is, the logic circuitry describing the program is scrambled in a way that a) for every input it still returns the same output as the original program would, and b) an attacker in possession of the obfuscated program is not able to learn anything about the original program, such as the involved key material. Therefore, the obfuscated program can be handed to a proxy, which uses it to perform re-encryption operations. However, the proxy is not able to extract the sender's private key from the scrambled circuit. Consequently, this makes the scheme *collusion-safe* and *non-interactive*. However, the construction by Hohenberger et al. is only *CPA-secure* and *not key-private*.

Ateniese, Benson, and Hohenberger [ABH08] were the first to define the property of key-private re-encryption keys. Informally, the key-private property says that an adversary is not able to distinguish the participants of a re-encryption key, while having access to all public keys and flexible interaction ability within the system. In their paper Ateniese, Benson, and Hohenberger also proposed an proxy re-encryption scheme fulfilling this *key-private* property. Furthermore, their construction is *unidirectional*, *single-hop*, *non-interactive*, and *collusion-safe*. However, their scheme is only *CPA-secure*.

While the previously discussed schemes were in the public-key setting, Green and Ateniese [GA07] addressed the problem of *identity-based* proxy re-encryption. In identity-based cryptography the public-key of an entity is represented by its identity. Therefore, identity-based proxy re-encryption transforms ciphertexts from one identity to another. Green and Ateniese proposed two schemes, which are *unidirectional* and *non-interactive*. Their first construction (called IBP1) is *multi-hop* and *CPA-secure*, whereas their second variant (called IBP2) is *single-hop* and *CCA-secure*. However, their constructions are *not collusion-safe*.

Chu and Tzeng [CT07] improved the scheme proposed by [GA07]. They presented an *identity-based* construction that does not rely on random oracles. Their scheme is *unidirectional*, *multi-hop*, *non-interactive*, and *CCA-secure*. However, it is *not collusion-safe*.

Xagawa and Tanaka [XT10] presented the first *lattice-based* proxy re-encryption scheme. Such a scheme uses lattices as its mathematical foundation, which, in contrast to previously used pairing based approaches, appears to be secure against cryptoanalysis with quantum computers. Their construction is *bidirectional*, *multi-hop* and *CPA-secure*. However, it is a) *not non-interactive*, since the private key of sender and receiver are required to generate a re-encryption key, and b) *not collusion-safe*, because if one private key used to generate the re-encryption key is known, the other private key can easily be recovered.

Aono et al. [Aon+13] proposed a *lattice-based* proxy re-encryption scheme, which is *unidirectional* and *multi-hop*. Furthermore, it improved on previous schemes by also being *CCA-secure* and *key-private*. However, during generation of re-encryption keys their algorithm relies on a tuple calculated from the recipient's private key. This reveals two issues. Firstly, this tuple has to be distributed beforehand to consider this proxy re-encryption scheme *non-interactive*. Secondly, if the proxy gets hold of this tuple, or if proxy and recipient collude, it is possible to recover the sender's private key. Therefore, the scheme of Aono et al. is *not collusion-safe*.

Nuñez, Agudo, and Lopez [NAL15] presented another *lattice-based* proxy re-encryption scheme. Their construction is *bidirectional*, *multi-hop* and very efficient, since it is based on the patented NTRU algorithm [HPS98]. However, their scheme also has some drawbacks, namely, it is *not collusion-safe*, *not non-interactive* and only *CPA-secure*.

In conclusion, this section described three main groups of proxy re-encryption schemes. The first constructions were pairing-based with gradually improved properties over their predecessors. Notable exceptions are the initial proxy re-encryption scheme by [BBS98] and the construction using obfuscation by [Hoh+07]. The second group consisted of identity-based schemes. Finally, the third group of schemes is based on lattices, which should provide security even against cryptographic analysis by quantum-computers. However, none of the evaluated constructions in this third group provided the property of collusion-safeness. A summary of the schemes and their properties is shown in table 4.1.

4.3. Property Requirements

This section explores the requirements demanded by the proposed identity broker system on the properties of proxy re-encryption schemes. For this purpose, we examine the individual properties of proxy re-encryption schemes, which were described in section 4.1. These properties are classified either as a requirement or as an optional but beneficial property. The resulting requirements are used in the following section 4.4 as basis for the evaluation of proxy re-encryption schemes.

Required Property: Unidirectional The proxy re-encryption scheme used in the proposed identity broker model has to be *unidirectional*. The following paragraphs first explain why an unidirectional scheme is more secure than a bidirectional scheme, while still having

	Unidirectional (U) Bidirectional (B)	Single-hop (S) Multi-hop (M)	Collusion-safe	Non-interactive	Identity-based	Lattice-based	CPA-secure (P) CCA-secure (C)	Key-private
Blaze, Bleumer, and Strauss [BBS98]	B	M	✗	✗	✗	✗	P	✗
Ateniese et al. [Ate+06]	U	S	✓	✓	✗	✗	P	✗
Canetti and Hohenberger [CH07]	B	M	✗	✗	✗	✗	C	✗
Libert and Vergnaud [LV08]	U	S	✓	✓	✗	✗	C	✗
Hohenberger et al. [Hoh+07]	U	S	✓	✓	✗	✗	P	✗
Ateniese, Benson, and Hohenberger [ABH08]	U	S	✓	✓	✗	✗	P	✓
Green and Ateniese [GA07], IBP1	U	M	✗	✓	✓	✗	P	?
Green and Ateniese [GA07], IBP2	U	S	✗	✓	✓	✗	C	?
Chu and Tzeng [CT07]	U	M	✗	✓	✓	✗	C	?
Xagawa and Tanaka [XT10]	B	M	✗	✗	✗	✓	P	?
Aono et al. [Aon+13]	U	M	✗	✓	✗	✓	C	✓
Nuñez, Agudo, and Lopez [NAL15]	B	M	✗	✗	✗	✓	P	?

Table 4.1.: This table summarizes the properties of the evaluated schemes. If a property is not stated clearly in the original or a referencing paper, this is denoted by a questionmark in the table.

the power to support re-encryption in both directions. Then, a paragraph describes why one direction of re-encryption is sufficient for the proposed use case. Finally, an attack is presented if the scheme would be bidirectional.

Firstly, in general, an unidirectional scheme is more secure than a bidirectional scheme, because the delegation of decryption rights is limited from one entity to another and does not have to be mutual. However, re-encryption in both directions can still be achieved by explicitly generating two unidirectional re-encryption keys in opposite directions. As a result, unidirectional schemes can be used to model the data sharing requirements of a particular use case more closely than bidirectional schemes, making the unidirectional property a more secure choice.

Secondly, in the particular use case of an identity broker system, re-encryption in one direction is sufficient. Data only has to be re-encrypted from ciphertext for the user to ciphertext for a particular service provider. If a service provider wants to provide encrypted data for the user, it can just use the publicly available public key of the user for encryption. Consequently, unidirectional re-encryption is sufficient for the proposed identity broker system.

Thirdly, a bidirectional, multi-hop re-encryption scheme would allow and attack on the proposed identity broker system. In the following example we assume that a user Alice has established a re-encryption key $rk_{a \leftrightarrow sp}$ with a legitimate service provider, and that there is a user Bob, who also has a relationship with the same service provider, resulting in $rk_{b \leftrightarrow sp}$. If the broker colludes with Bob, they could perform the attack explained in

the following four steps. At the beginning, the broker retrieves Alice's encrypted data C_a. Then, the broker re-encrypts C_a with $rk_{a \leftrightarrow sp}$ for the service provider, resulting in C_{sp}. Subsequently, the unintended direction of $rk_{b \leftrightarrow sp}$ is used to re-encrypt C_{sp} for Bob, giving C_b. Finally, together they would be able to read all of Alice's encrypted data by decrypting C_b with Bob's private key.

In conclusion, for the use in the proposed identity broker system, an *unidirectional* re-encryption scheme is required. This decision is motivated by three facts. Firstly, in the use case, re-encryption is only used in one direction. Secondly, unidirectional schemes are more secure, as only minimal power is released. Finally, a bidirectional scheme would enable an attack in the identity broker use case if the scheme would also be multi-hop.

Optional Property: Single-Hop For the use with the proposed identity broker system, the *single-hop* property should be preferred, but neither the single-hop nor the multi-hop property are a requirement. The following two paragraphs describe why single-hop schemes are sufficient for this particular use case, and explain why single-hop schemes could provide security benefits but essentially have the same functionality as multi-hop schemes.

Firstly, in this particular use case, being able to re-encrypt once is sufficient. The user's ciphertext only has to be re-encrypted for one of many service providers. For the proposed identity broker system no functionality was defined that would benefit from further re-encryption.

Secondly, in general, a single-hop scheme can provide security benefits over a multi-hop scheme. Since the single-hop property limits the re-encryption powers, an unintended re-encryption key from the receiver to a potentially untrusted third party cannot be used. Other than that, single-hop and multi-hop schemes provide the same functionality from the receiver's point of view. In the single-hop case, further re-encryption can still be achieved by decrypting the ciphertext and encrypting the plaintext again for the next party.

As a result, neither the single-hop nor the multi-hop property are required. However, a *single-hop* scheme should be preferred, since such a scheme could prevent a proxy from making unintended use of additional re-encryption keys.

Required Property: Collusion-Safe The proxy re-encryption scheme selected for the proposed identity broker system has to be *collusion-safe*. The following two paragraphs explain why a collusion-safe re-encryption scheme is more secure in general, and describe the threat posed by a not collusion-safe scheme to the proposed identity broker system.

In general, a collusion-safe scheme is more secure, since it prevents a semi-trusted proxy, that colludes with a re-encryption recipient, from recovering the sender's private key.

In the identity management use case, the user's private key could be recovered from a re-encryption scheme that is not collusion-safe if an adversary gains access to two keys. Firstly, the adversary has to obtain a re-encryption key from the user to a service provider. Secondly, the adversary has to acquire the private key material of that service provider. If the broker colludes with a service provider, for which the user has generated a re-encryption key, thogether they have access to the required key material. Therefore, they could reconstruct the user's private key if the re-encryption scheme is not collusion-safe.

As a result, the proposed identity broker system requires a *collusion-safe* re-encryption scheme, since this property protects the user's private key from collusion of the broker with a service provider.

Required Property: Non-Interactive The proxy re-encryption scheme used in the proposed identity broker system has to be *non-interactive*, since the generation of a re-encryption key should only involve the user's private key and the service provider's public key. Otherwise, if the scheme would not be non-interactive, private keys of both the user and the service provider would be needed to generate a re-encryption key. Therefore, user and service provider would have to reveal their private keys to a fully trusted third party. As such trust relationships from multiple re-encryption participants to one specific external entity are hard to establish, such an entity was not defined in the proposal. In conclusion, the selected proxy re-encryption scheme is required to be *non-interactive*, so that re-encryption keys can be generated by the user alone, who just needs to obtain the service provider's public key aside from her own private key.

Required Property: Not Identity-Based The proposed identity broker system requires its proxy re-encryption scheme to be *not identity-based*. The following three paragraphs first describe the general benefit of identity-based cryptography. Secondly, a paragraph explains why this benefit would be of no consequence in the proposed identity broker system. Finally, it is pointed out why the additional infrastructure required by an identity-based re-encryption scheme would actually be a disadvantage.

In general, identity-based cryptography provides a benefit. A string representing the identity of the recipient can be used to derive her public key. Therefore, it is not necessary to obtain the recipient's certificate to acquire an authentic copy of her public key.

For the particular use case in an identity broker system, the general benefit of not requiring certificates does not add any value. An authentic public key would be required in three scenarios. Firstly, a user requires her own public key to encrypt for herself. The user is in possession of her own public key anyway, and therefore does not have to obtain the key by certificate or identity-based derivation. Secondly, the service provider's public key is required by the broker to generate the re-encryption key via an external cryptographic service. The need to generate a re-encryption key arises while the broker communicates with the service provider. Therefore, the broker can just obtain the service provider's certificate via this channel. Thirdly, the service provider might want to encrypt data for the user and thus requires the user's public key. In this case, the service provider has already been associated with the user via the broker. Therefore, the service provider is able to obtain the user's certificate from the broker, where it was deposited. As a result, all scenarios in an identity broker system can be realized with certificate-based trust, and consequently an identity-based scheme would not add any value.

In the proposed identity broker system, an identity-based scheme would actually be a disadvantage. Identity-based cryptography relies on a third party to generate private keys and issue the key material to participants. This third party knows all private keys and consequently has to be fully trusted by all involved participants. Introducing such a powerful third party requires more trust from the participants than relying on a certificate authority issuing certificates. The reason is that the certification authority does not learn

the participant's private key, and therefore is not as powerful, which in turn reduces the required trust. As a consequence, the trust requirements for an identity-based scheme are higher than for a certificate-based system.

As a result, the selected proxy re-encryption scheme has to be *not identity-based*. The benefit of not having to distribute certificates is of no consequence, as it can easily be accommodated in the identity protocol. On the contrary, an identity-based scheme would require additional, fully trusted, infrastructure. This would complicate the trust relationships considerably and was therefore not intended in the proposed identity broker system.

Optional Property: Lattice-Based The proxy re-encryption scheme selected for the proposed identity broker system does not have to be lattice-based, but a *lattice-based* scheme would add value and should therefore be preferred in the selection process. Thus far, no algorithm for cryptographic analysis of lattice-based primitives has yet been discovered that could harness the benefits of quantum computers. In consequence, a scheme based on lattices would be more secure than traditional cryptography. As the development of quantum computers is an ongoing research topic, protection against such cryptographic analysis would be desirable in the long term. In conclusion, *lattice-based* schemes would be beneficial, but are currently not a requirement.

Required Property: CPA-secure In order to protect the user's privacy, the proxy re-encryption scheme selected for the proposed identity broker system is required to be *CPA-secure*. The following two paragraphs describe the general issue caused by schemes that are not CPA-secure, and explain why such a not CPA-secure scheme would be a privacy concern in the proposed identity broker system.

In general, a scheme that does not fulfill the CPA-secure property can be a security problem. There are two prerequisites to perform a chosen-plaintext attack. Firstly, the adversary requires access to an encryption oracle, which provides ciphertexts for arbitrarily chosen plaintexts. Secondly, the adversary has to be able to obtain ciphertexts that should be analyzed. If those prerequisites are fulfilled, for a not CPA-secure scheme, an adversary is able to infer information about the contents of an obtained ciphertext or link multiple related ciphertexts.

For the use case in an identity broker system, a re-encryption scheme that does not fulfill the CPA-secure property would be an issue for the user's privacy. In this scenario, the two prerequisites to perform a chosen-plaintext are satisfied. Firstly, access to an encryption oracle is universally given, as public keys are available and they can be used to construct ciphertexts of arbitrary plaintexts. Secondly, a man in the middle between user and service provider has access to ciphertext generated by the user. Assuming the communication channels are secure, only intermediary parties could serve as adversaries. Those parties include curious brokers or attribute providers, since they deal with encrypted attributes of users. As the prerequisites are fulfilled, an adversary would be able to attack a not CPA-secure scheme, and thereby learn something about the user's encrypted attributes. For example, in a deterministic and therefore not CPA-secure scheme, an adversary could encrypt an age value and examine where the same ciphertext appears. As a result, a proxy re-encryption scheme that is not CPA-secure does not appropriately protect the user's

privacy.

In conclusion, the proxy re-encryption scheme selected for the proposed identity broker system has to be *CPA-secure*, as a scheme that does not fulfill this property would compromise the user's privacy.

Optional Property: CCA-secure The scheme selected for the proposed system should be *CCA-secure*, however this is not a requirement. In the next paragraph, the general issue caused by schemes that are not CCA-secure is described. Then, we provide details on why only the broker could perform a chosen-ciphertext attack in the proposed system. The following paragraph explains why the impact of a not CCA-secure scheme is very limited.

In general, schemes that are not CCA-secure could be a security problem. Consider an attacker who is able to modify a ciphertext C to get another related and valid ciphertext C'. This attacker would decrypt C' to get a related plaintext P', which can be used to learn something about the original plaintext P of C. Consequently, the attacker could use this approach to gather information about one challenge ciphertext of the game defining CCA-security and therefore win. However, if the scheme is CCA-secure, a modification of the ciphertext that allows to reason about the original plaintext cannot exist.

In the proposed identity broker system, only the broker would fulfill the prerequisites to perform a chosen-ciphertext attack. One requirement for a chosen-ciphertext attack is the availability of an decryption oracle. Only two actors of the proposed system perform decryptions and are therefore possible candidates. Firstly, the user decrypts attributes for herself but does not provide and feedback regarding the outcome of this operation to an outside observer and therefore cannot serve as an decryption oracle. Secondly, service providers decrypt the user's attributes which were obtained from the broker. As the broker ensures the authenticity of the provided attributes, an external attacker is not able to query this decryption oracle. Only this broker would be able to use the service provider as decryption oracle.

However, the impact of a not CCA-secure scheme is very limited. Even though there is no proof that it is impossible to modify a ciphertext to obtain a related ciphertext, two approaches can severely limit an attacker's ability to exploit schemes that are not CCA-secure. Firstly, by limiting the service provider's feedback, for example in case of an error, this decryption oracle only provides minimal information to the attacker. Secondly, by monitoring the decryption results of ciphertexts sent by the broker, the service provider could realize the broker's malicious behavior.

In conclusion, even though the proxy re-encryption scheme selected for the proposed identity broker system is not required to be *CCA-secure*, such an property would be beneficial.

Optional Property: Key-Private The proxy re-encryption scheme selected for the proposed identity broker system should preferably be *key-private*, even though this property is not a requirement. The following three paragraphs first describe the general privacy enhancing effect of key-private schemes. Then, a paragraph explains how the key-private property would prevent an adversary from linking users and services based on key material in the proposed identity broker system. Finally, a reason is presented why the key-private

property has no consequences in the proposed identity broker system due to usability measures.

In general, key-private re-encryption keys enhance the privacy of participants in a proxy re-encryption scheme. If the used scheme would not be key-private, an adversary would be able to identify the involved parties of a re-encryption key, when given access to their public keys. In contrast, key-private re-encryption keys do not reveal information about their participants, thus protecting the privacy of the involved parties.

The proposed identity broker system would benefit from a key-private re-encryption scheme, since such a scheme would prevent an adversary from linking users with services they use based on key material. In order to exploit a not key-private scheme, two prerequisites have to be fulfilled. Firstly, an adversary has to get access to the public keys. As those keys are publicly available, this prerequisite is fulfilled. Secondly, an adversary has to obtain re-encryption keys, which should be analyzed regarding their participants. For example, a curious broker would be in possession of all re-encryption keys and therefore satisfy the second prerequisite. With those prerequisites fulfilled, an adversary could exploit a not key-private scheme to identify the participants of a re-encryption key. As users generate a re-encryption key for services they use, the adversary would be able to link users to these services.

Due to necessary usability improvements, the key-private property is of no consequence in the proposed identity broker system. In order to minimize user interaction, the broker compiles profiles, which store all decisions the user made regarding her connection to a service provider. Those profiles include data such as the selected identity provider and information about attribute routing. Without those profiles the user would be confronted with a very repetitive work-flow, which would deteriorate the user experience considerably. Therefore, these profiles are a requirement in order to enhance usability. However, such profiles already link users and services they use. Therefore, the key-private property is of no consequence and does not provide additional value.

As a result, the selected proxy re-encryption scheme is not required to be *key-private*, since this property is not able to enhance privacy due to required usability measures. However, a re-encryption scheme with this property should be preferred, since it prevents linking of users and services that is purely based on re-encryption key material.

In conclusion, this section identified five required properties as well as four optional properties, which could still be beneficial. The re-encryption scheme used in the proposed identity broker system is required to be *unidirectional*, *collusion-safe*, *non-interactive*, *not identity-based* and *CPA-secure*. In addition to the required properties, there are some optional properties, which would add value to the re-encryption scheme. Those optional properties of a re-encryption scheme include being *single-hop*, *lattice-based*, *CCA-secure* and *key-private*.

4.4. Suitable Re-Encryption Schemes

This section evaluates proxy re-encryption schemes, described in section 4.2, with regards to the requirements defined in section 4.3. Table 4.2 lists the individual re-encryption schemes and

illustrates which properties are fulfilled.

	Required					Optional			
	Unidirectionality	Collusion-safe	Non-interactive	Not Identity-based	CPA-secure	Single-hop	Lattice-based	CCA-secure	Key-private
Blaze, Bleumer, and Strauss [BBS98]	✗	✗	✗	✓	✓	✗	✗	✗	✗
Ateniese et al. [Ate+06]	✓	✓	✓	✓	✓	✓	✗	✗	✗
Canetti and Hohenberger [CH07]	✗	✗	✗	✓	✓	✗	✗	✓	✗
Libert and Vergnaud [LV08]	✓	✓	✓	✓	✓	✓	✗	✓	✗
Hohenberger et al. [Hoh+07]	✓	✓	✓	✓	✓	✓	✗	✗	✗
Ateniese, Benson, and Hohenberger [ABH08]	✓	✓	✓	✓	✓	✓	✗	✗	✓
Green and Ateniese [GA07], IBP1	✓	✗	✓	✗	✓	✗	✗	✗	?
Green and Ateniese [GA07], IBP2	✓	✗	✓	✗	✓	✓	✗	✓	?
Chu and Tzeng [CT07]	✓	✗	✓	✗	✓	✗	✗	✓	?
Xagawa and Tanaka [XT10]	✗	✗	✗	✓	✓	✗	✓	✗	?
Aono et al. [Aon+13]	✓	✗	✓	✓	✓	✗	✓	✓	✓
Nuñez, Agudo, and Lopez [NAL15]	✗	✗	✗	✓	✓	✗	✓	✗	?

Table 4.2.: This table summarizes the properties of the evaluated algorithms with regards to the requirements. Properties are grouped into required and optional properties. The four schemes that completely fulfill the requirements are highlighted. If a property is not stated clearly in the original or a referencing paper, this is denoted by a questionmark in the table.

The following four schemes completely fulfill the requirements of the proposed identity broker system: 1) [Ate+06], 2) [LV08], 3) [Hoh+07], as well as 4) [ABH08]. In addition, all of those schemes satisfy the preferred single-hop property. Further beneficial properties are provided by two schemes, namely the scheme proposed by [LV08] is CCA-secure, and the scheme presented by [ABH08] is key-private. As a result, from a theoretical point of view, the schemes by [LV08] and [ABH08] would be the most favorable.

However, from a practical point of view, the scheme presented by Ateniese et al. [Ate+06] was chosen for the implementation of the identity broker system, as it reduces the development effort, while completely satisfying the requirements. From the four schemes fulfilling the requirements, only the scheme by [Ate+06] has a prototypical implementation available. This implementation could be used as basis for the development of re-encryption-related subcomponents of the proposed system.

It is worth mentioning that lattice-based algorithms would be of interest in the future. The scheme by [Aon+13] almost fulfilled the requirements and it further proved to be lattice-based, CCA-secure and key-private. Sadly, it does not fulfill the requirement of collusion-safeness. However, further research in this direction may result in similar proxy re-encryption schemes that meet the requirements, while providing all those other beneficial properties.

In conclusion, this section evaluated proxy re-encryption schemes with regard to the identified property requirements. The four schemes by [Ate+06], [LV08], [Hoh+07], and [ABH08] completely fulfill the requirements. Out of those four schemes the construction by [Ate+06] was chosen for the implementation of the proposed identity broker system, as it considerably reduces the development effort.

4.5. Chapter Conclusion

In this chapter, multiple proxy re-encryption schemes have been evaluated in a general survey and for the use in the proposed identity broker system. Proxy re-encryption schemes can hold a multitude of properties, which describe their applicability possibilities and security characteristics. In a general survey, multiple proxy re-encryption schemes have been examined regarding their properties. To select a proxy re-encryption scheme for the proposed identity broker system, property requirements have been identified. The scheme is required to be unidirectional, collusion-safe, non-interactive, not identity-based, and CPA-secure, while optional properties include being single-hop, lattice-based, CCA-secure and key-private. Those requirements have served as the basis in the evaluation of suitable schemes. The four schemes by [Ate+06], [LV08], [Hoh+07], and [ABH08] completely fulfill the requirements. While LV and ABH provide favorable optional properties, AFGH reduces the development effort, and, therefore, has been chosen for the implementation of the proposed identity broker system.

5. Evaluation of Emulated Re-Encryption

This chapter compares the general concept of proxy re-encryption with emulated re-encryption, where a ciphertext is transformed by consecutive decryption and encryption. Those types of re-encryption can be analyzed based on multiple aspects, including properties defined for proxy re-encryption, standardization, and trust relationships. Since emulated and proxy re-encryption differ not only in the capabilities they provide but also in the requirements they place on their environment, there is an impact on the deployment possibilities.

This chapter is organized as follows: First, in section 5.1, emulated re-encryption is described in more detail. Then, section 5.2 relates emulated re-encryption with the properties of proxy re-encryption described in section 4.1. Subsequently, in section 5.3, the ability to use standardized underlying cryptography is explained as an advantage of emulated re-encryption over proxy re-encryption. Section 5.4 explores the trust relationships required by emulated re-encryption and further describes the differences in comparison to proxy re-encryption. In section 5.5, the deployment options of re-encryption key generation and re-encryption operations are compared between emulated and proxy re-encryption. Finally, this chapter is concluded in section 5.6

5.1. Definition of Emulated Re-Encryption

In emulated re-encryption, a fully trusted proxy performs the re-encryption operation by first decrypting the ciphertext for one entity and subsequently encrypting the obtained plaintext for another entity. Diagram 5.1 illustrates such an emulated re-encryption process. In this scenario, the proxy is the entity that performs the consecutive de- and encryption. Since proxy re-encryption does not explicitly specify which key material has to be provided by the recipient of the re-encrypted ciphertext, there are two possible definitions for the re-encryption key, which differ in the involved key material. This differentiation has an impact on the reasoning in the following sections. The two definitions for re-encryption keys are: a) The re-encryption key is represented by the tuple (sk_a, pk_b), where the private key sk_a is used to decrypt the original ciphertext and the public key pk_b is used to encrypt the obtained plain message for the other entity. b) The re-encryption key is represented by the tuple (sk_a, sk_b). As before, the private key sk_a is used to decrypt the original ciphertext into its underlying message. Assuming the public key can be generated from the respective private key, the public key pk_b derived from sk_b is used to encrypt the message for the other entity.

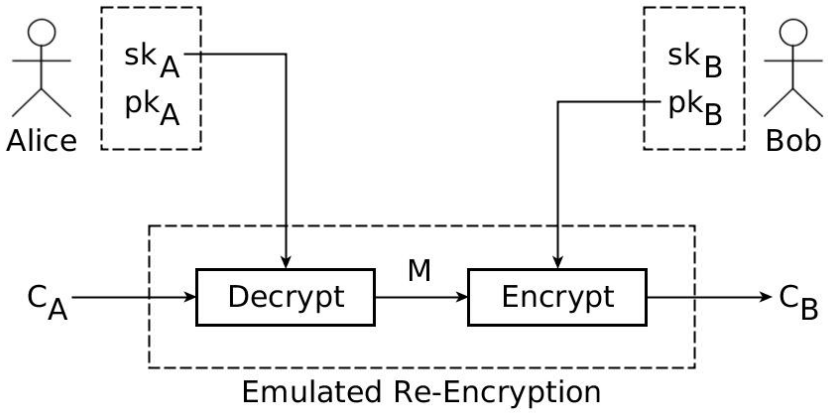


Figure 5.1.: This diagram outlines the process of emulated re-encryption, where a ciphertext C_A obtained by encrypting the plaintext message M for Alice is transformed into a ciphertext C_B for Bob. This re-encryption is performed by first decrypting C_A with Alice's private key sk_A into the underlying message M. Subsequently, this message M is encrypted for Bob with his public key pk_B, giving C_B.

5.2. Properties

The detailed definition of emulated re-encryption provied in the previous section enables us to reason about emulated re-encryption with regard to the properties of proxy re-encryption. As we will see, some properties rely on the underlying encryption scheme, while others depend on one of the two re-encryption key definitions. The following paragraphs examine the individual properties.

Unidirectional or Bidirectional: Whether an emulated re-encryption scheme is uni- or bidirectional depends on the used re-encryption key definition. The following two paragraphs describe both cases.

If the re-encryption key $rk = (sk_a, pk_b)$ consists of a private and a public key, the scheme is *unidirectional*. As the private key has to be used for decryption and the public key has to be used for encryption, the re-encryption is limited to one direction. Therefore, re-encryption keys containing both a private and a public key make the scheme *unidirectional*.

However, if the re-encryption key $rk = (sk_a, sk_b)$ includes two private keys, the emulated re-encryption scheme is *bidirectional*. Assuming the public key can be derived from its corresponding private key, either of the private keys can be used for decryption, whereas the public key for encryption is derived from the respective other private key. A re-encryption key with two private keys allows to perform re-encryption in both directions, making the scheme *bidirectional.*

Multi-Hop: An emulated re-encryption scheme is always *multi-hop*. A re-encrypted ciphertext does not differ structurally from an initial ciphertext, as the same encrypt operation was involved in both cases. Therefore, it is possible to further re-encrypt an already re-encrypted ciphertext. As a result, emulated re-encryption schemes are *multi-hop*.

Not Collusion-Safe: Emulated re-encryption is *not collusion-safe*. In both definitions of the re-

encryption key, (sk_a, pk_b) and (sk_a, sk_b), the proxy has direct access to the private key of the entity for which the original ciphertexts are encrypted. Consequently, the proxy does not even have to collude with another entity to obtain that private key material. As a result, emulated re-encryption is certainly *not collusion-safe*.

Non-Interactive or Not Non-Interactive: Depending on the used re-encryption key definition, the emulated re-encryption scheme can be non-interactive. The following two paragraphs describe both cases.

If the re-encryption key $rk = (sk_a, pk_b)$ consists of a private and a public key, the scheme is *non-interactive*. As only the public key pk_b of the delegatee is involved, which is publicly available, she does not have to participate in the re-encryption key generation. Therefore, the delegator is able to generate re-encryption keys containing her private key and the delegatee's public key by herself, making the scheme *non-interactive*.

However, if the re-encryption key $rk = (sk_a, sk_b)$ includes two private keys, the emulated re-encryption scheme is *not non-interactive*. Since the delegatee does not want to reveal her private key to the delegator, this delegator is not able to generate a re-encryption key by herself without interaction. Therefore, re-encryption keys consisting of two private keys make the scheme *not non-interactive*.

Not Key-Private: Emulated re-encryption schemes are *not key-private*. Regardless of the used re-encryption key definition, an adversary who is given access to all public keys is able to identify the participants of a re-encryption key. The adversary simply has to examine the keys contained in the re-encryption key. A contained public key can be matched to one of the given public keys. For a contained secret key, the adversary is able to find the corresponding public key by encrypting arbitrary data with all given public keys and checking which decryption with the secret key leads to the same data. Thus, an adversary with access to the re-encryption key and all public keys is able to identify the participants of that re-encryption key, making the emulated re-encryption scheme *not key-private*.

Possibly Identity-Based: An emulated re-encryption scheme *can be identity-based* if the underlying encryption scheme is also identity-based. In such an identity-based setting, a public key is represented by publicly known data identifying the recipient, for example by an email address. Such identity information is placed into the re-encryption key as public key, which is then used in the encryption operations. Therefore, it is possible to realize an emulated re-encryption scheme that is identity-based by using identity-based encryption.

Possibly Lattice-Based: Emulated re-encryption schemes *can be lattice-based*. If a lattice-based encryption scheme is used in the construction of the emulated re-encryption scheme, this emulated re-encryption scheme is considered to be lattice-based.

Possibly CPA-Secure: Depending on the underlying encryption scheme, emulated re-encryption *can be CPA-secure*. If the used encryption scheme is CPA-secure, then the emulated re-encryption scheme is also CPA-secure.

Possibly CCA-Secure: As described for the CPA-secure property, the emulated re-encryption scheme *can be CCA-secure* depending on the encryption scheme used in the construction of the emulated scheme. If the used encryption scheme is CCA-secure, the resulting emulated

re-encryption scheme is also CCA-secure.

	Unidirectional (U) Bidirectional (B)	Single-Hop (S) Multi-Hop (M)	Collusion-Safe	Non-Interactive	Identity-Based	Lattice-Based	CPA-Secure (P) CCA-Secure (C)	Key-Private
$rk = (sk_a, pk_b)$	U	M	✗	✓	?	?	?	✗
$rk = (sk_a, sk_b)$	B	M	✗	✗	?	?	?	✗

Table 5.1.: This table summarizes the properties of the emulated re-encryption, which depend on the re-encryption key definition. Properties that can be achieved with an appropriate underlying encryption scheme are marked with a questionmark.

As a result of this property assessment, emulated re-encryption schemes are *multi-hop*, *not collusion-safe*, and *not key-private*, whereas the other properties either depend on the definition of the re-encryption key or on the underlying encryption technology. On the one hand, in case the re-encryption consists of a private and a public key, the emulated re-encryption scheme is *unidirectional* and *non-interactive*. On the other hand, if the re-encryption key contains two private keys, the emulated re-encryption scheme is *bidirectional* and *not non-interactive*. Depending on the properties of the underlying encryption scheme, the emulated scheme can be *CPA-secure*, *CPA-secure*, *identity-based*, and *lattice-based*. The properties of emulated re-encryption are summarized in table 5.1.

5.3. Standardization

In this section, emulated and proxy re-encryption are compared regarding their level of standardization and the impact caused by different levels is analyzed. Use cases for re-encryption usually involve multiple parties, as different entities encrypt, re-encrypt and decrypt the data. In order to enable interactions between those parties, they have to find a common understanding of the involved algorithms as well as the exchanged in- and output data. Such a common understanding can be provided by a standard. The following paragraphs first examine the level of standardization of proxy re-encryption schemes. Subsequently, the positive impact caused by standardized algorithms used in emulated re-encryption is described. In addition, this section discusses remaining challenges regarding the integration of standardized algorithms used in emulated re-encryption into other standards.

Proxy re-encryption lacks standardization and is therefore not widely integrated into other specifications. Over the last two decades, a multitude of proxy re-encryption schemes have been proposed. However, to the best of my knowledge, only the IEEE P1363.3 workgroup [IEE13] attempted to standardize two early identity-based schemes. This lack of standardization is a major cause why proxy re-encryption is not widely integrated into other specifications. Therefore, representations for key materials and exchanged data have to be defined for each

application individually. Additionally, other security mechanisms do not support proxy re-encryption but have to be extended. As a result, the use of proxy re-encryption in a complex system with multiple communicating components is a work-intensive challenge.

Emulated re-encryption can be realized using traditional encryption schemes, which are standardized and well established. This provides multiple advantages. Already provisioned key material, which is for example stored on a smart card, can be reused. In addition, those standardized algorithms are often supported by other higher level standards. For instance, container formats, such as CMS [Hou09] or JWT [JBS15b], build on traditional encryption schemes. As a result, emulated re-encryption does not require as much effort to integrate as proxy re-encryption.

However, even though emulated re-encryption can be used with many standard encryption schemes, it is not possible to replace traditional encryption in every situation. Consider a container format such as JWT. In such a case, emulated re-encryption works well with the hybrid encryption mechanism. However, if the whole container is integrity-protected after the encryption was applied, for example by wrapping a signature or message authentication code around it, subsequent re-encryption would invalidate this protection. Nevertheless, if the integrity of the container's payload was first protected and the whole container was then encrypted, re-encryption is still possible. Therefore, it is still necessary to carefully analyze the composition of cryptographic primitives in order to determine the applicability of emulated re-encryption.

In conclusion, proxy re-encryption lacks standardization, which is a main cause why it is not widely used. In contrast, emulated re-encryption uses standardized encryption schemes and, therefore, is easy to integrate into existing solutions, where already provisioned key material and standardized security mechanisms can be used. However, it is not always possible to replace traditional encryption schemes with emulated re-encryption schemes, as the concept of re-encryption clashes with the goal of integrity preserving mechanisms.

5.4. Trust Relationships

While the generation of re-encryption keys has the same trust implications for both emulated and proxy re-encryption due to the same input parameters, the required trust in the proxy performing the re-encryption operation is different. The trust requirements further depend on the used re-encryption key definition. In the following two paragraphs, the re-encryption key generation and, subsequently, the re-encryption operation are examined.

For the re-encryption key generation, trust is required from participants that provide a private key and, therefore, depends on the re-encryption key definition. In case the re-encryption key is created from a private and a public key, only the delegator has to fully trust the entity generating the re-encryption key. Consequently, the delegator is able to generate the re-encryption key herself. However, if the re-encryption key is created from two private keys, both owners, the delegator and the delegatee, have to fully trust the re-encryption key generator. As neither participant wants to reveal her private key, the generation has to be performed by a trusted third party.

The re-encryption operation can be performed by a semi-trusted third party in a proxy re-encryption scheme, whereas emulated re-encryption requires complete trust in the proxy from all participants that placed their private keys in the re-encryption key. For proxy re-encryption, delegator and delegatee only have to trust the proxy to perform correctly. They do not have to concern themselves with curious proxies which try to inspect intermediate calculation, as proxy re-encryption schemes guarantee that the underlying message is not exposed in the re-encryption process. Therefore, a proxy only has to be semi-trusted for proxy re-encryption. In case an emulated re-encryption scheme is used, participants that provided their private key in the re-encryption key generation have to fully trust the proxy, as this proxy is able to extract their key material from the tuple representing the re-encryption key.

	Re-Encryption Key Generator		Re-Encryption Proxy	
	Delegator	Delegatee	Delegator	Delegatee
Re-Encryption Key from sk_a and pk_b				
Proxy Re-Encryption	F	-	S	S
Emulated Re-Encryption	F	-	F	S
Re-Encryption Key from sk_a and sk_b				
Proxy Re-Encryption	F	F	S	S
Emulated Re-Encryption	F	F	F	F

Table 5.2.: This table compares the trust relationships between, on the one side, delegator or delegatee and, on the other side, re-encryption key generator or proxy. The trust relationship can be not-relevant (-), semi-trusted (S) or fully trusted (F).

In conclusion, emulated re-encryption has higher trust requirements than proxy re-encryption, as private key material contained in the re-encryption key is handled directly at the proxy. Table 5.2 summarizes the trust relationships between, on the one side, delegator or delegatee and, on the other side, re-encryption key generator or proxy.

5.5. Comparison of Deployment Options

This section compares the deployment options of components implementing emulated and proxy re-encryption functionality in the proposed identity broker system. Such a comparison showcases the evaluation process and highlights differences between emulated and proxy re-encryption in a real-world example. The two operations of interest are the generation of re-encryption keys and the re-encryption of ciphertexts. The following paragraphs begin by describing possible deployment locations with their capabilities. Then, influencing factors that serve as selection criteria are explained. Finally, an evaluation of the deployment options is presented for the re-encryption key generation and the re-encryption operation in both emulated and proxy re-encryption schemes.

The cryptographic operations can be deployed in three locations. These operations can be performed by the broker directly. Alternatively, a cryptgraphic service in the cloud or on the

user's phone can implement such operations and offer them to the broker via a cryptographic service interoperability layer (CrySIL). Those environments offer different capabilities, which are summarized in table 5.3.

Broker: An identity broker in the cloud is only semi-trusted, as the infrastructure is not under the control of the user and the user does not fully trust the operator of the cloud resources. However, the cloud provides multiple benefits, which include scalability and availability. As cloud services are scalable, a high number of requests can be process in short time. Also, no user intervention is required, as cloud services are constantly available.

CrySIL Node in the Cloud: A CrySIL node in the cloud can be fully trusted, as the user selects a node at some provider that holds this property. In contrast to the cloud-based broker, this trust can be achieved because the user trusts the cloud operator to follow high security standards, such as employing a hardware security module. Additionally, a cloud node enjoys the same benefits as the identity broker. It offers high performance and is always available to process requests without requiring user interaction.

CrySIL Node on the User's Phone: If a user does not completely trust a cloud-based service, her own mobile phone can provide cryptographic operations. As the user is in complete control of her phone, it is fully trusted. In addition, modern phones provide enough computing power to complete infrequent cryptographic operations. However, a phone might not always be available to respond to requests. For example, the phone might not be connected to a data network or it might simply be turned off. In those situations, user interaction may be required to make the phone available to perform cryptographic requests.

	Trust	Performance	Always Available
Broker	S	H	✓
CrySIL Node in the Cloud	F	H	✓
CrySIL Node on the User's Phone	F	L	✗

Table 5.3.: This table compares the capabilities of different deployment locations for cryptographic operations required in the proposed identity broker system. The environment can either be semi-trusted (S) or fully-trusted (F). The performance is denoted low (L) or high (H).

The deployment options of cryptographic operations are limited by the trust requirements, the frequency of the operation, and the interaction with the user. The *trust* requirements, which have been explained in the previous section 5.4, represent a major selection criterion for the deployment location. For the generation of re-encryption keys, the trust requirements are the same in both emulated and proxy re-encryption, but the re-encryption operation places different demands on the trust into the execution environment. In addition, the *frequency* of the operation and the *interaction* with the user also play an important role. The generation of a re-encryption key happens rarely and always involves the user. In contrast, the re-encryption has to be performed frequently, whenever a service provider requires access to the user's data, and does not involve interaction with the user, as consent was given beforehand.

The deployment criteria for the re-encryption key generator and the re-encryption operation for both the emulated and the proxy re-encryption case are defined in the following paragraphs. In order to identify possible deployment locations three factors, namely trust requirements, frequency of execution, and user interaction, are taken into consideration. The resulting selection criteria are summarized in table 5.4.

Deployment Criteria for the Re-Encryption Key Generator: The three deployment criteria for the re-encryption key generator are the same for both emulated and proxy re-encryption. Firstly, as explained in the previous section, the re-encryption key generation has to be performed in an execution environment that is fully trusted by all participants supplying their private key material for both emulated and proxy re-encryption. In the proposed identity broker system, the re-encryption key is generated from a user's private and a service provider's public key. Consequently, only the user has to fully trust the environment executing the re-encryption key generator, which is represented by a CrySIL node. Secondly, the re-encryption key generation is rarely performed, as this operation is only triggered when a new service provider is registered. Thirdly, the registration of a service provider involves the user, therefore user interaction is possible.

Deployment Criteria for the Re-Encryption Operation: This paragraph examines three criteria for the deployment of the re-encryption operation. Firstly, with emulated re-encryption, the re-encryption operation has to be performed in a fully trusted environment. In contrast, with proxy re-encryption a semi-trusted execution environment is sufficient. Secondly, the re-encryption operation is executed frequently, as it is triggered every time the service provider requests a user's attribute. Thirdly, the user is not involved in the re-encryption process, and therefore user interaction is not available.

	Trust	Frequency	Interaction
Re-Encryption Key Generation			
Emulated Re-Encryption	F	L	✓
Proxy Re-Encryption	F	L	✓
Re-Encryption			
Emulated Re-Encryption	F	H	✗
Proxy Re-Encryption	S	H	✗

Table 5.4.: This table compares selection criteria for deployment locations of re-encryption key generator and re-encryption proxy with both emulated and proxy re-encryption. The operation can either require a semi-trusted (S) or fully-trusted (F) environment. The frequency is denoted low (L) or high (H).

The deployment options for the individual operations can be identified by evaluating the possible deployment locations with regards to the defined deployment criteria. In the following paragraphs, first the deployment options of the re-encryption key generator and, subsequently, of the re-encryption operations are explored. The results of this evaluation are presented in table 5.5.

Deployment Options for the Re-Encryption Key Generator: The re-encryption key generator can be deployed regardless of the underlying emulated or proxy re-encryption scheme, as both schemes have identical deployment criteria. Since the key generator has to be

executed in a fully trusted environment, the broker is the only inappropriate deployment location. The low number of re-encryption key generation operations can be handled by both low and high performance services. Therefore, CrySIL nodes in the cloud as well as on the user's phone are suitable. Finally, a cloud-based CrySIL node is always available, whereas a node on the user's phone can be made accessible, due to the user's availability for manual intervention. As a result, re-encryption key generators can be deployed on both CrySIL nodes, regardless of the underlying emulated or proxy re-encryption scheme.

Deployment Options for the Re-Encryption Operation: Based on the defined deployment criteria, it is possible to identify deployment options for the re-encryption operation. In general, a CrySIL node on the user's phone is not suitable to perform the re-encryption operation due to two reasons. The phone might not be available and no user is present to intervene in the case of a re-encryption operation. In addition, a low-power phone might not be able to keep up with the high number of re-encryption requests. As the broker as well as the CrySIL node in the cloud offer high performance, the number of requests does not represent an obstacle for them. The deployment options of the re-encryption operation differ based on the use of emulated or proxy re-encryption. Emulated re-encryption requires full trust in the environment executing the re-encryption, which eliminates the broker as deployment option. In contrast, the re-encryption operation of a proxy re-encryption scheme only requires a semi-trusted environment, which is fulfilled by both remaining deployment locations, namely the broker and the CrySIL node in the cloud. As a result, according to the above identified criteria, the re-encryption operation of a emulated re-encyrption scheme can only be deployed on a CrySIL node in the cloud. In contrast, the re-encryption operation of a proxy re-encryption scheme can be deployed on the CrySIL node in the cloud as well as on the broker.

	Broker	CrySIL Node in the Cloud	CrySIL Node on the User's Phone
Re-Encryption Key Generation			
Emulated Re-Encryption	✗	✓	✓
Proxy Re-Encryption	✗	✓	✓
Re-Encryption			
Emulated Re-Encryption	✗	✓	✗
Proxy Re-Encryption	✓	✓	✗

Table 5.5.: This table summarized the deployment options that satisfy the deployment criteria for the re-encryption key generator and the re-encryption operation when using emulated or proxy re-encryption.

In conclusion, for the proposed identity broker system there are multiple criteria that impact the possible deployment locations of cryptographic operations. Trust plays a major role in this selection process. Especially the high trust requirements of emulated re-encryption limit the deployment options. As the user's phone is under the complete control of the user, it enjoys a very high level of trust. This makes the phone a favorable execution environment. However, when considering other criteria such as performance and availability, a phone's applicability fares poorly in comparison with a cloud-based CrySIL node or broker.

5.6. Chapter Conclusion

In this chapter, proxy re-encryption has been compared to emulated re-encryption, where a ciphertext is transformed by first decrypting the ciphertext and then encrypting it for another recipient. Such a emulated re-encryption scheme can be characterized with properties originally defined for proxy re-encryption. It is multi-hop, not collusion-safe and not key-private. Depending on the used re-encryption key definition, the scheme can either be unidirectional and non-interactive or bidirectional and not non-interactive. Other properties such as CPA-secure, CCA-secure, identity-based, and lattice-based rely on the underlying encryption scheme. In addition, as emulated re-encryption employs standardized encryption schemes, integration with existing protocols and reuse of already provisioned key material is possible. In contrast, proxy re-encryption suffers from a lack of standardization. However, emulated re-encryption requires a higher level of trust in the environment performing the re-encryption operations, as private key material is involved. Especially those high trust requirements limit the deployment options of emulated re-encryption. That is, emulated re-encryption cannot be performed on the broker, since it is only semi-trusted by the user.

6. Implementation

This chapter describes the implementation of the proposed identity broker system, where multiple components interact to complete a multi-step process in which the user is authenticated and requested attributes are retrieved. In order to realize such a broker system, an identity broker, a cloud-based CrySIL node, and a service provider were implemented. In addition, a library that simplifies the integration into service providers was designed. Also, the implemented system makes use of existing identity and attribute providers. Figure 6.1 gives an overview of the involved components and their connections.

This chapter is organized as follows: Section 6.1 describes the individual steps of the authentication and attribute acquisition process. Then, in section 6.2, the user experience during this process is explained based on the example of an online copyshop. Subsequently, section 6.3 provides a detailed description of the implemented components. Finally, a conclusion of this chapter is presented in section 6.4.

6.1. Process Steps

The process of authentication and attribute acquisition requires interaction between multiple components. These interaction steps are illustrated as a sequence diagram in figure 6.2. The remainder of this section describes the process flow, which can roughly be divided into the following steps: 0) relationships and key material are set up, 1) authentication and attribute acquisition are delegated to the broker, 2) a profile containing the user's decisions is created, 3) the user is authenticated at an identity provider, 4) access is gained to the user's preferred attribute providers, 5) attributes are re-encrypted for the service provider, 6) authentication codes are exchanged for tokens, 7) further attributes are retrieved, and, finally, 8) the user's initial is answered.

0. **Setup:** In order to take part in the proposed identity broker system, relationships between components have to be established and multiple keys have to be provisioned. The following two paragraphs describe the prerequisites for user, broker and service provider, first with regard to the relationships and then concerning the key material.

   Multiple relationships between the components and actors of the implemented broker system have to be established. The user has to associate with one or more identity and attribute providers. Then, the user informs her identity broker during registration of her accounts on these identity and attribute providers. In addition, the user sets up or registers at a trustworthy CrySIL node and provides connection data to the broker. Furthermore, trust has to be established between service provider and broker, as well as between broker and both identity and attribute provider.

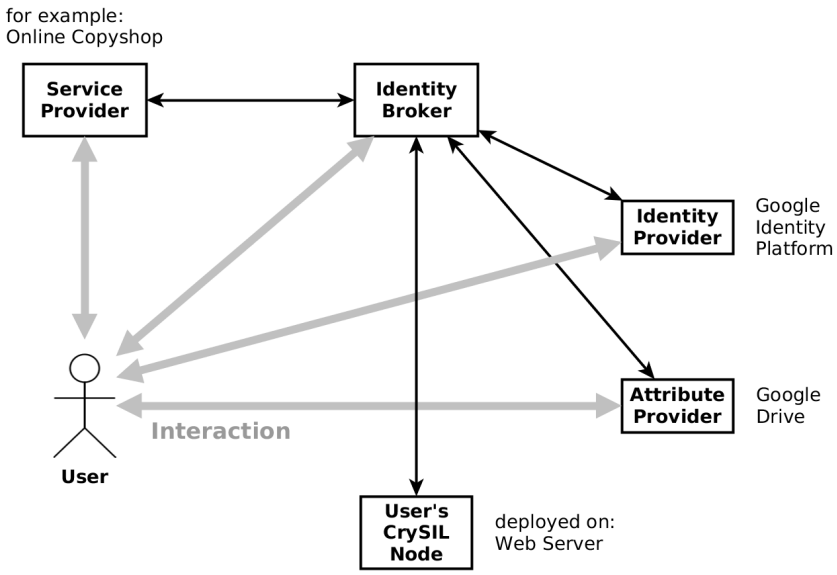


Figure 6.1.: This diagram illustrates the components of the implementation as well as their connections. A service provider, for example an online copyshop, delegates the authentiaction of users and the acquisition of required attributes to the identity broker. This broker authenticates the user via Google as identity provider and retrieves the requested attributes from Google Drive as attribute provider. The re-encryption key material is generated on the user's CrySIL node, which is deployed on a web server.

In order to perform cryptographic operations, key material is required by user, broker, and, finally, service provider. Firstly, the user has to generate a traditional signature key pair $(sk_{u,sign}, pk_{u,sign})$ as well as encryption key material for the proxy re-encryption scheme $(sk_{u,enc}, pk_{u,enc})$. The signature private key $sk_{u,sign}$ is used to sign the user's attributes, and, subsequently, the signed attribute is encrypted for the user with the encryption public key $pk_{u,enc}$. The user then uploads the signed and encrypted attributes to her attribute providers. Secondly, the broker requires a re-encryption key, which is generated on demand in a subsequent process step. In addition, the broker also generates a signature key pair $(sk_{b,sign}, pk_{b,sign})$, which is used to sign a set of retrieved and re-encrypted attributes. Thirdly, the service provider generates encryption key material for the proxy re-encryption scheme $(sk_{sp,enc}, pk_{sp,enc})$. With the contained private key $sk_{sp,enc}$, the service provider is able to decrypt the user's re-encrypted attributes.

The actual process of authentication and attribute acquisition consists of eight steps. The previously described step sets up the prerequisites for the interactions in the system. If a new user or service provider joins the system, parts of that setup step have to be performed. Therefore, this step is executed only infrequently. The following paragraphs describe the process steps required for authentication and attribute acquisition.

1. **Delegation to Identity Broker:** The user points her web browser to a protected resource at the service provider, which requires users to authenticate and to share some attributes in order to fulfill the requests. As the service provider does not want to implement the authentication and attribute acquisition, it delegates this task to the user's identity broker. The user's preferred broker is determined through OpenID Connect Discovery. To delegate

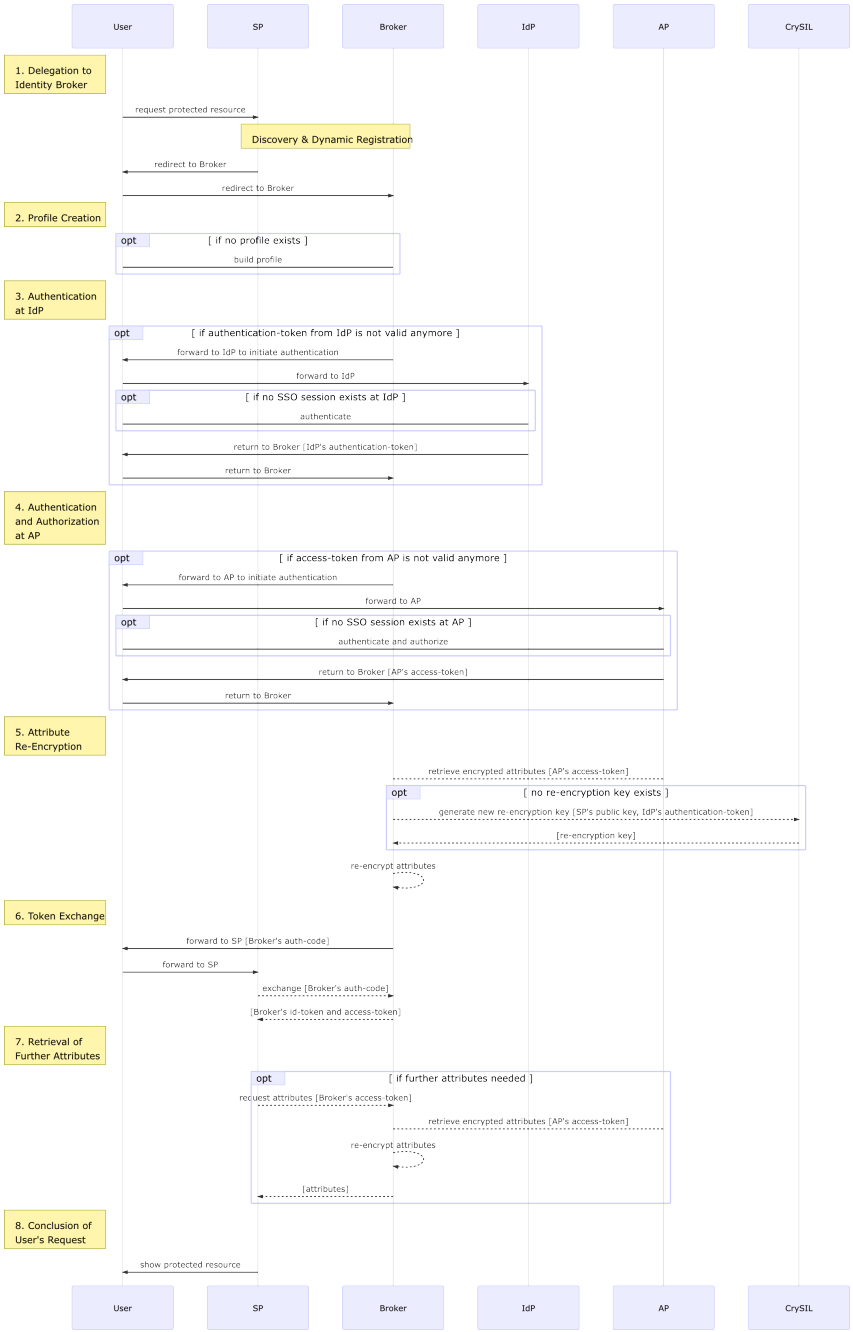


Figure 6.2.: This sequence diagram illustrates the individual process steps required to authenticate and retrieve the user's data. The process can be divided into eight steps. 1) The service provider delegates authentication and attribute acquisition to the identity broker, which 2) builds a profile about the user, 3) authenticates the user at an identity provider, and 4) obtains authorization to access the user's attribute provider. 5) Then, the broker retrieves the user's encrypted attributes, generates a re-encryption key via the user's CrySIL node, and re-encrypts the attributes for the service provider. 6) The service provider has to exchange its authentication code for tokens, 7) which can be used to request further attributes. 8) Finally, the service provider is able to fulfill the user's initial request.

tasks to the user's broker, this broker has to be registered with the service provider. An ad-hoc association is achieved through OpenID Connect Dynamic Registration. Both, discovery and dynamic registration, are explained in further detail in section 2.2. Once the service provider and the broker are introduced to each other, the service provider delegates user authentication and attribute acquisition by redirecting the user's browser to her identity broker. This redirect also carries the requested permissions as well as the service provider's public key.

2. **Profile Creation:** Once the user's browser arrives at the identity broker, the user has to make decisions on how to handle the service provider's request. These decisions cover three aspects. Firstly, the user is shown which service provider requires authentication and access to which attributes and, subsequently, is asked for consent. Secondly, the user has to select an identity provider, where she wants to authenticate. Thirdly, the user has to specify which attributes should be shared as the requested attributes. Those decisions are stored as a profile describing the relationship between user and service provider. Therefore, this step only has to be carried out once and can be skipped in subsequent runs.

3. **Authentication at Identity Provider:** The identity broker delegates authentication to an identity provider. The required process steps depend on the particular identity provider. In the following, typical steps for the integration of an external identity provider are examined. Then, optimizations regarding the user interactions involved in this process flow are discussed.

   In order to authenticate the user via an identity provider, multiple protocol steps have to be performed. This paragraph describes a simplified flow that typical identity protocols, such as OpenID Connect and SAML, have in common. The identity broker integrates the Google Identity Platform [Goo15b], which offers an identity provider that supports OpenID Connect, and, therefore, fits this typical flow. Firstly, the broker redirects the user's browser to the identity provider. Then, this identity provider authenticates the user through whatever means. For example, Google verifies knowledge of username and password. After successful authentication, the identity provider issues some kind of authentication token, such as a JSON Web Token, and redirects the user's browser back to the broker. Finally, the broker verifies the authentication token.

   The above described authentication process requires user interaction, which can be optimized to provide a more user-friendly experience. There are two scenarios in which no user interaction has to take place. If the broker is in possession of a valid authentication token from a recent authentication process, this step can be skipped altogether. Otherwise, even after the user has been redirected to the identity provider, authentication information from a previously opened SSO session at the identity provider could be used to skip further authentication and the usually involved user interaction. As a result, user interaction can be omitted if information about a recent authentication is reused.

4. **Authentication and Authorization at Attribute Provider:** In order to fulfill the service provider's request, the identity broker requires access to the user's encrypted attributes, which reside at the attribute provider. To gain access to these attributes the identity provider requests authorization, which usually also requires users to authenticate. The following two paragraphs describe a simplified process flow to obtain authorization as

well as usability optimizations that allow to skip steps.

The broker integrates attribute providers to accesses the user's attributes. The steps required to obtain authorization depend on the particular attribute provider. This paragraph describes a simplified version of the OAuth [Har12] protocol, which is also used by Google Drive [Goo15a]. Firstly, the broker redirects the user's browser to the attribute provider with a set of requested access permissions. The attribute provider usually authenticates the user in order to reach an authorization decision regarding the requested permissions. During this authentication and authorization user interaction is required. After successful authorization, the user's browser is redirected back to the broker. This redirect also carries an access-token, which is used by the broker in subsequent requests for attributes to demonstrate authorization.

In two scenarios, the user interaction required for this authorization process can be bypassed. If the user's session at the broker still contains a valid access token for the attribute provider, the whole authorization process can be skipped. Otherwise, the broker has to redirect the user to the attribute provider for authentication. However, in case the attribute provider opened a SSO session for the user, the authentication and potentially also the authorization can be skipped at the attribute provider. Instead the user is immediately redirected back to the identity broker with a new access token. As a result, user interaction can be skipped if an access token is still available or if the attribute provider makes use of SSO sessions.

5. **Attribute Re-Encryption:** In order to share the user's attributes with a service provider, the broker has to obtain the encrypted attributes from attribute providers and, subsequently, re-encrypt them for the service provider. The user stores encrypted attributes at attribute providers. These attributes are encrypted by the user for herself to be able to decrypt them on demand. In order to share these encrypted attributes, they have to be re-encrypted for the service provider. With proxy re-encryption, the broker can act as proxy and perform the re-encryption. However, this re-encryption requires a re-encryption key that can be generated from the user's private key material. The following paragraphs first describe the process steps required to share attributes and examine possible optimizations.

   The broker collects encrypted attributes from the user's selected attribute providers and re-encrypts them with a re-encryption key. Firstly, the broker retrieves the user's encrypted attributes from the attribute providers according to the information stored in the user's profile. In the retrieval request, the broker demonstrates authorization by including a previously obtained access token. Secondly, the broker communicates with the user's CrySIL node to generate a re-encryption key. This key generation involves the user's private key and, therefore, requires authorization. The requirements to grant access are specified in a user-defined policy. If the data provided in the request satisfies the policy, a new re-encryption key is generated for the service provider and sent back to the broker. Finally, the broker uses the re-encryption key to re-encrypt the retrieved attributes for the service provider.

   The steps in the above described process can be optimized in some scenarios. By storing a once generated re-encryption key alongside the user's profile, the key generation can be skipped in subsequent process runs. However, during the first run, the broker has to

gain authorization to involve the user's private key in the re-encryption key generation. Depending on the user's access policy, this authorization could be granted without user interaction. For example, the policy might be satisfied with a valid authentication token issued by a trusted identity provider and a service provider's certificate proving the validity of the service provider. As a result, even during the first process run, a re-encryption key could be generated without interacting with the user if her access policy allows it.

6. **Token Exchange:** As described in OpenID Connect, the broker issues an authentication code, which has to be exchanged by the service provider for the actual authentication and access token. When redirecting the user's browser back to the service provider, the broker only includes an authentication code in the request. The authentication token and access token are therefore not exposed to the user's browser. Instead, the service provider exchanges the authentication code for the broker's authentication and access token. The authentication token contains information about the delegated authentication as well as the user's re-encrypted attributes. These re-encrypted attributes can be decrypted with the service provider's private key. The access token can be used later on by the service provider to demonstrate authorization when accessing the broker's API.

7. **Retrieval of Further Attributes:** In some scenarios, the service provider might require further attributes. For example, in an online copyshop, the user might first authenticate, but later on have to provide pictures and shipping as well as payment information. Therefore, the service provider sends a request for further attributes with the access token previously obtained from the broker. If the original consent granted by the user does not cover the new request, the service provider has to request further permissions and retry. Otherwise, the broker uses the still valid access token from the attribute provider to retrieve the required attributes. With the stored re-encryption key, the broker re-encrypts the encrypted attributes for the service provider. Finally, these re-encrypted attributes are returned. As a result, the service provider is able to decrypt the obtained re-encryption attributes.

8. **Conclusion of the User's Request:** After the service provider's requirements have been fulfilled, the initial request by the user for a protected resource can be fulfilled. Authentication has been ensured through the broker's authentication token, as the broker is trusted to only issue such a token after successful authentication at an identity provider. The user's required attributes have been supplied through the authentication token and by requesting further attributes. Therefore, the service provider has been ensured of the user's authenticity and has obtained all required attributes. As a result, the service provider is finally able to respond to the user's initial request.

In conclusion, the process can be divided into the following steps. 0) First, relationships between components are set up and key material is provisioned. 1) The service provider delegates authentication and attribute acquisition to the identity broker, which 2) builds a profile about the user, 3) authenticates the user at an identity provider, and 4) obtains authorization to access the user's attribute provider. 5) Then, the broker retrieves the user's encrypted attributes, generates a re-encryption key via the user's CrySIL node, and re-encrypts the attributes for the service provider. 6) The service provider has to exchange its authentication code for tokens,

7) which can be used to request further attributes. 8) Finally, the service provider is able to fulfill the user's initial request.

6.2. User Experience

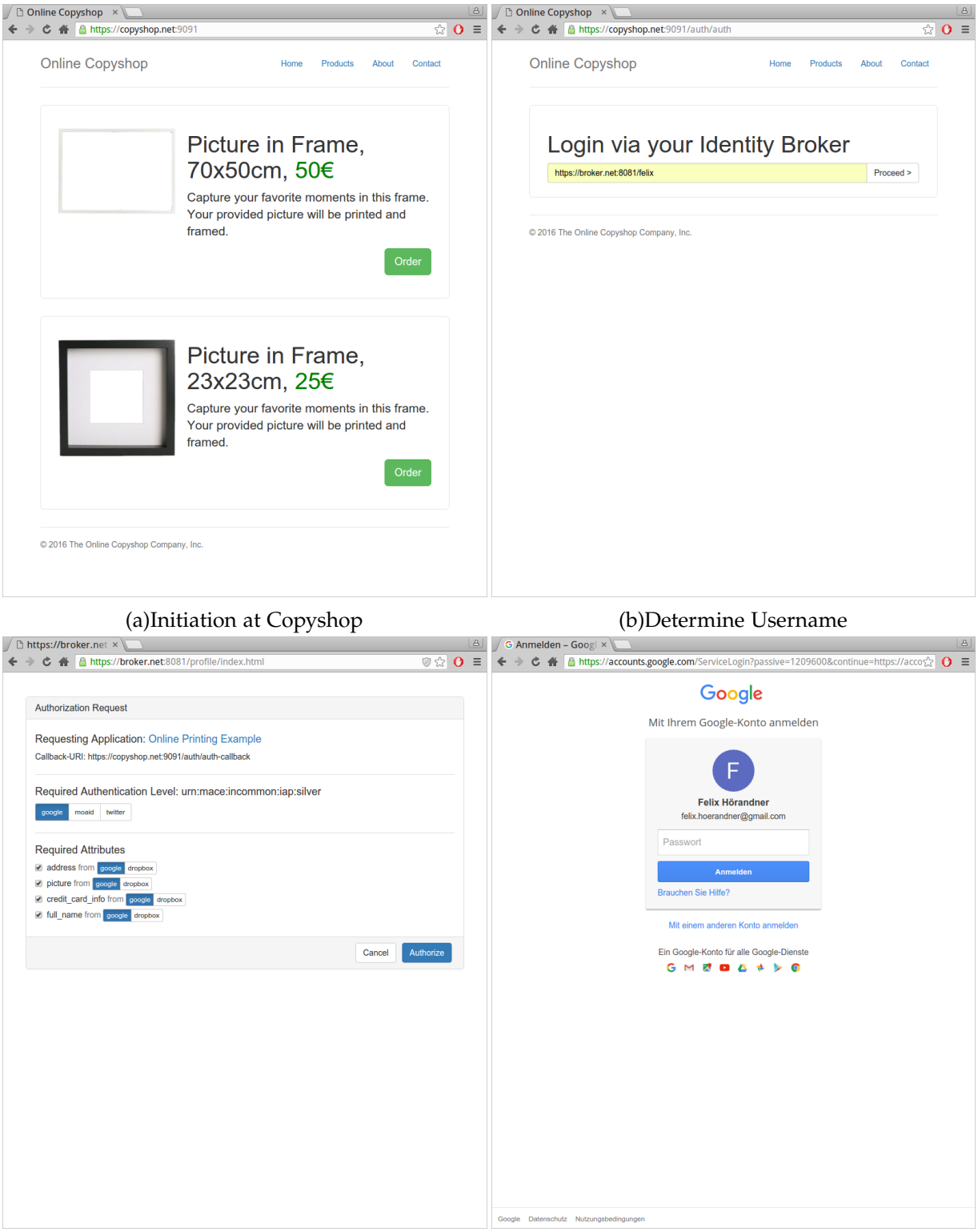
In contrast to the technical process steps described above, this section examines the implementation of the proposed identity broker system from the user's perspective. An online copyshop was implemented as a service provider (SP) to highlight the user experience in a practical environment. In this copyshop example, a user wants to order a printed and framed version of a picture provided by her. This process requires both authentication at Google as identity provider (IdP), and the user's data from Google as an attribute provider (AP), including a shipping address, billing information, as well as the picture. This section is related to the general user experience analysis in section 3.3. The first part of this section explaines the individual process steps involving the user. Then, typical scenarios are examined and usability improvements are discussed.

The user interacts with some components of the identity broker system, during the authentication and attribute acquisition process triggered in the copyshop example. In the following, the individual steps in the full process are described in further detail. An overview of these steps is presented in figure 6.3, whereas full-size screenshots can be found in appendix A. Note that optimizations to this rather lengthy process for subsequent requests are presented afterwards.

1. **Initiate at Copyshop:** In the copyshop example, the user wants to get one of her pictures printed and framed. Figure 6.3a illustrates the copyshop service provider as it offers different options for frames to the user. Once the ordering process is started, the service provider requires authentication and some data from the user. The required data includes her full name, a shipping address, billing information, as well as the picture that should be printed. To provide users with a convenient and secure way to provide these data, the service provider integrates with the proposed identity broker system.

2. **Determine Username:** In order to delegate the authentication and data acquisition to an identity broker, this broker has to be discovered first. As defined in OpenID Connect, the discovery of a user's preferred broker is based on the hostname included in the user's username. Therefore, the service provider asks for the username, which is shown in figure 6.3b. Once the user's preferred broker is discovered and its configuration is loaded, the service provider can be registered at this broker. Then, the service provider forwards the user's browser to the identity broker.

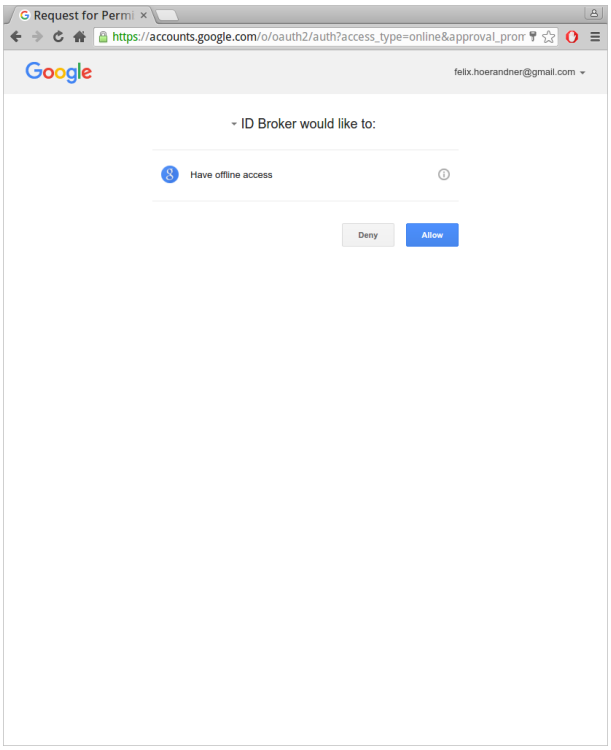
3. **Select IdP and APs:** As illustrated in figure 6.3c, the broker presents information about the service provider's request to the user. This request includes the desired authentication level, as well as the required attributes. Based on this information, the user can either cancel the request or provide consent by selecting an identity provider and suitable attribute providers. These selections make up the basis of a profile which stores the user's decisions regarding a service provider.



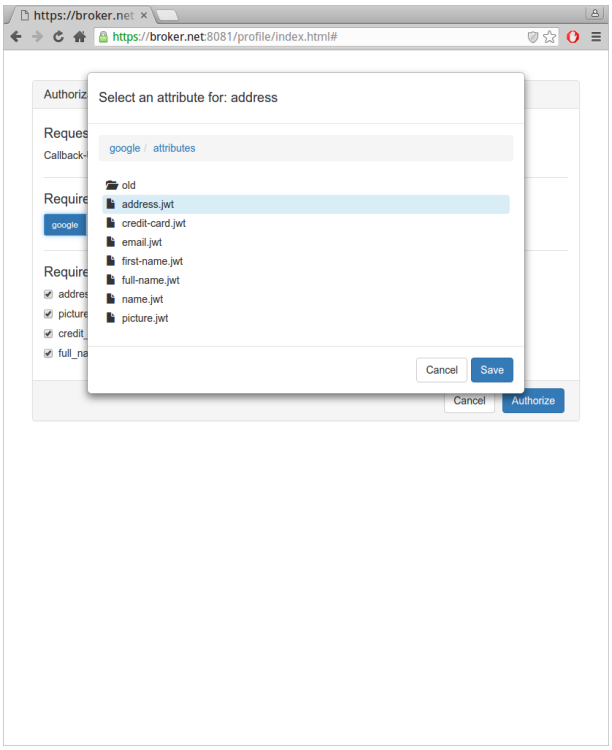
(a)Initiation at Copyshop (b)Determine Username

(c)Select IdP and APs (d)Authenticate at Google

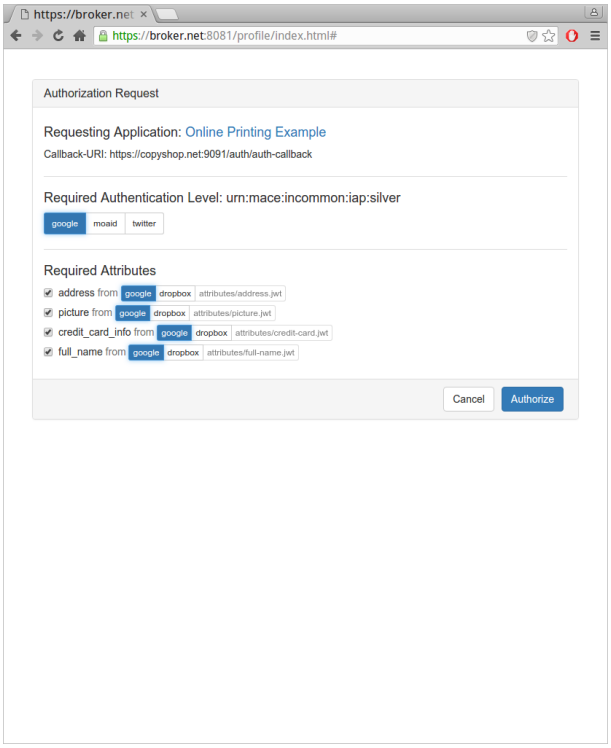
Figure 6.3.: User experience in the copyshop example.



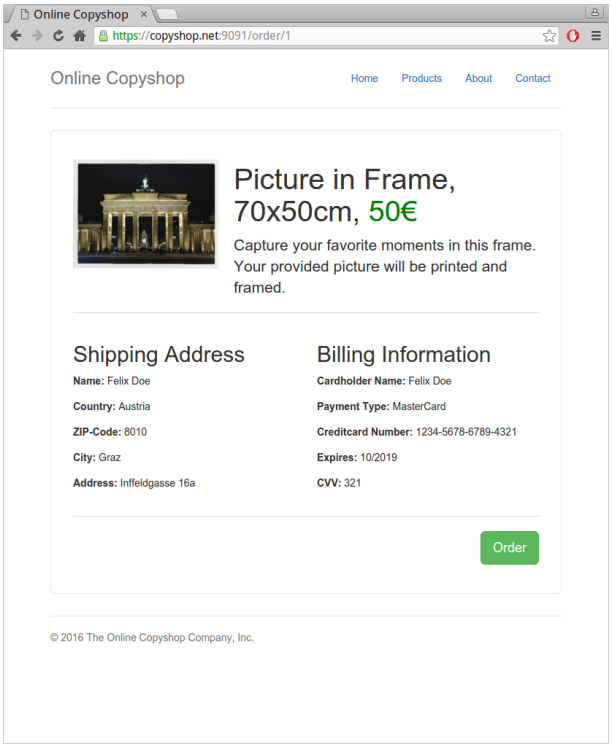
(e)Authorize at Google



(f)Select Attributes to Share



(g)Grant Consent



(h)Conclusion at Copyshop

Figure 6.3.: User experience in the copyshop example. (continued)

4. **Authenticate at Google:** After the user selected an identity provider, her browser is redirected there in order to authenticate. In the implementation of the proposed identity broker system, Google was integrated as identity provider. Figure 6.3d shows the interface presented to the user for authentication based on username and password.

5. **Authorize at Google:** When using Google as identity provider the authentication does not also represent implicit authorization as well. Therefore, the user has to provide explicit consent, as depicted in figure 6.3e.

(skipped) **Authentication and Authorization at the AP:** After completing the process at the identity provider, authentication and authorization also have to be performed at the attribute provider. However, as the implementation of the broker system uses Google Drive as attribute provider, the before obtained authentication and authorization from Google as an identity provider is sufficient. As a result, this step can be skipped in this process flow.

6. **Select Attributes to Share:** At the broker, the user is asked to select which data items at an attribute provider should be returned for the individual requested attributes. As this selection process is implemented by browsing the available encrypted JWT files at Google Drive, authorization to access the file index has to be obtained beforehand. The attribute selection process is shown in figure 6.3f. Again, the user's decisions are added to the profile stored at the broker for later reuse.

7. **Grant Consent:** Once the user has completed the attribute routing, the compiled information regarding the service provider's request is presented to her. This is shown in figure 6.3g. After the user was able to review the decisions, she finally grants consent. A re-encryption key is generated at the user's CrySIL node, without her interaction. This key material is then used to re-encrypt the encrypted attributes.

8. **Conclusion at Copyshop:** In the end, the service provider receives the authentication information and re-encrypted attributes. By decrypting the attributes, they can be used in the copyshop ordering process. Figure 6.3h shows the copyshop's ordering page. There, the name, shipping address and billing information is listed. Furthermore, the user's originally encrypted picture is shown as a preview. As a result, the service provider is able to obtain the required data with minimal effort, while the user is presented with a convenient way to share her stored information.

Multiple optimizations can be applied to the above process, as already described in section 3.3. These optimization are based on the reuse of data compiled during a previous run. Therefore, in subsequent runs, it is possible to cut down the number of steps requiring user interaction, which improves usability. The following paragraphs explain which steps can be skipped in three typical scenarios. A summary of the necessary steps in these scenarios is presented by table 6.1.

First Authentication: During the first time a user uses a service provider with the identity broker, interaction is required in all of the above mentioned steps, assuming the user does not have a SSO session at Google. In this scenario, all the required information has to be entered by the user. Firstly, the user has to provide the service provider with her username. Then, the user has to make decisions, including selecting an IdP and AP as well as routing

the attributes. In addition, authentication and authorization at Google is required. Finally, the user explicitly grants consent, in order to stay in control of the data sharing process and to be able to apply some changes.

Subsequent Authentication with Single Sign On Sessions: In a scenario, where the user has already used the service with the identity broker and a SSO session is available at Google, most of the steps requiring interaction can be skipped. The service provider requires the username, which can be offered by a long-term cookie in some situations. However, from a usability point of view, it might not be wise to skip the step of determining the username completely. In case the device's current user is not the same as the user who performed the initial authentication process, the username should be changeable. One way to achieve this would be to pre-fill the username in the form, which only requires the user to confirm if the name is correct. Since the user's profile already contains the user's previous decisions, it should not be necessary to select an IdP and AP or change the attributes to share. Furthermore, authentication and authorization at Google can be skipped if a SSO session is available. Finally, the user has to grant consent. At this point, the user is able to adapt previous decisions.

Subsequent Authentication without Single Sign On Sessions: In another typical scenario, the user has already used the service with the identity broker, but does not possess a currently valid SSO session. This scenario differs from the last in only one aspect. The user has to re-authenticate at Google and perform a re-authorization of the broker.

	First Authentication	Subsequent Authentication with SSO Sessions	Subsequent Authentication without SSO Sessions
Initiate at Copyshop	✓	✓	✓
Determine Username	✓	∼	∼
Select IdP and APs	✓		
Authenticate at Google	✓		✓
Authorize at Google	✓		✓
Select Attributes to Share	✓		
Grant Consent	✓	✓	✓
Conclusion at Copyshop	✓	✓	✓

Table 6.1.: This table summarizes three typical scenarios and the steps requiring user interaction. The first time a user uses a service provider with the identity broker, all steps have to be performed, assuming the user does not have an open SSO session at Google. In subsequent runs of the authentication process, many steps can be skipped by reusing decisions and information previously provided by the user. Note that the tilde sign (∼) represents a step that can be partially skipped.

In conclusion, the user experience includes interaction in multiple steps for the copyshop example. After initiation of the process, the service provider has to determine the user's username, in order to discover the identity broker. At the broker, the user as to make decisions, including which IdP and AP should be used, as well as, which the attributes should be shared. Authentication and authorization is delegated to Google. In the end, the user grants final consent and, as a result, the copyshop service receives the required data. This rather lenghty procedure only has to be performed the first time a user uses a service with the identity broker. In subsequent runs of this process, most steps requiring user interaction can be skipped by reusing previous decisions.

6.3. Components

The implementation of the proposed identity broker system consists of multiple components. Service providers offer services that might require users to authenticate and provide access to their attributes. This authentication and attribute acquisition is performed by the user's preferred identity broker. The broker outsources authentication to identity providers, such as the Google Identity Platform, and retrieves the user's data from attribute providers, such as Google Drive. In order to share the user's data, the broker requires re-encryption keys. These keys are generated by the user's CrySIL node. Such a node can be deployed on the user's phone or run on a trusted server.

The following subsections provide a detailed explanation of the individual components, which includes block diagrams of the architecture as well as descriptions of subcomponents. The remainder of this section is organized as follows: Firstly, section 6.3.1 describes the service provider, which protects some of its resources with a protection library. Then, section 6.3.2 presents the broker's subcomponents, which provide an OpenID Connect interface, delegate tasks to identity and attribute providers, communicate with the user's CrySIL node to generate a re-encryption key, and re-encrypt attributes. Finally, in section 6.3.3, the user's CrySIL node is described. Neither identity nor attribute providers are explored in detail, as these components are external entities that only have been integrated.

6.3.1. Service Provider

In order to offer its service, the service provider requires users to authenticate and to share data. Such a service provider consists of multiple sub-components, which are illustrated in figure 6.4. The following paragraphs describe 1) the types of resources, 2) the protection library, 3) an OpenID Connect client, 4) the JWT library, and, finally, 5) a proxy re-encryption library.

The service provider presents multiple *resource* endpoints to the *user's browser* to offer its service. Some of these resources might require the user to authenticate and to share data. For example, an online copyshop needs a user 1) to authenticate, in order to link her with her orders, and 2) to provide access to data such as the documents to print, the delivery address or billing information.

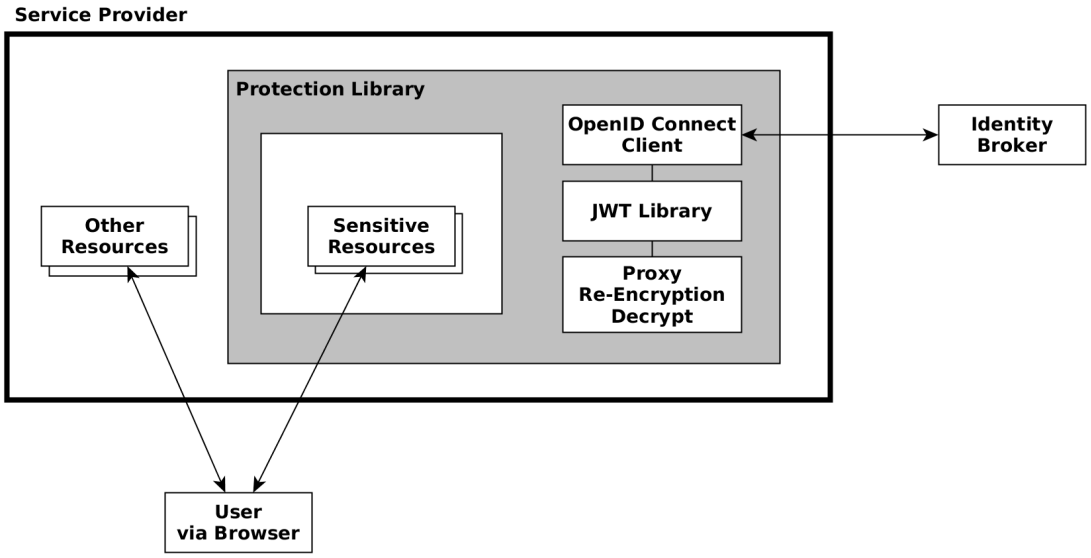


Figure 6.4.: This diagram illustrates the subcomponents of a service provider. The service provider offers multiple resources to the user's browser. Some of these resources required authentication and access to attributes. Therefore, the protection library wraps around them. This library includes an OpenID Connect client, which communicates with the identity broker, a JWT library, which handles tokens containing attributes, as well as a proxy re-encryption library, which decrypts the re-encrypted attributes.

The *protection library* enforces these requirements for authentication and access to data. This library wraps around resources and offers a convenient API to require authentication and access to the user's attributes. Internally, this protection layer can be divided into three subcomponents.

The *OpenID Connect client* interacts with the identity broker. This client handles discovery and dynamic registration. Once an authentication and attribute acquisition process is started, the client forwards the user to the broker, exchanges authentication code for tokens and requests further attributes.

The *JWT library* handles tokens and attributes, as both are in JWT format. As the result of the authentication process, the broker issues an id-token, which has to be verified. Furthermore, the user's attributes are re-encrypted and provided as a JSON web token. Therefore, this library verifies the token's signature and triggers decryption.

Finally, the decryption of re-encrypt attributes is performed by the *Proxy Re-Encryption Library*. This cryptographic library is a common component used not only by the service provider but also by the identity broker and the user's CrySIL node. As evaluated in chapter 4, the proxy re-encryption scheme by Ateniese et al. [Ate+06] was implemented.

In conclusion, some of the resources offered by the service provider might require authentication and access to the user's attributes. Therefore, these resources are wrapped by the protection library, which can be divided into three subcomponents. Firstly, a OpenID Connect client interacts with the identity broker. Secondly, the JWT library handles tokens as well as the user's attributes. Finally, the common re-encryption library decrypts the re-encrypted attributes.

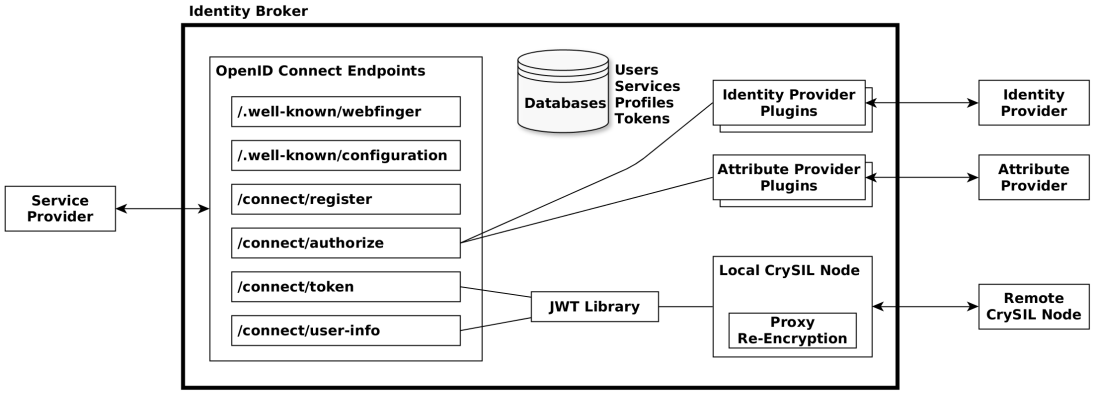


Figure 6.5.: This diagram illustrates the subcomponents of the identity broker. Multiple OpenID Connect endpoints interact with service providers to enable discovery, dynamic registration and delegation of user authentication and attribute acquisition. Authentication of the user and authorization to access her attributes is implemented by identity and attribute provider plugins. Obtained tokens containing the user's attributes are handled by a JWT library. This library re-encrypts the attributes for a service provider by using the services provided by the local CrySIL node. This local node communicates with a remote CrySIL node that is trusted by the user, in order to generate required re-encryption keys from the user's private key material.

6.3.2. Identity Broker

The identity broker offers to service providers both user authentication as well as access to the user's attributes. The implemented broker consists of multiple subcomponents, which are illustrated in figure 6.5. The following paragraphs describe 1) OpenID Connect endpoints, 2) databases, 3) plugins for different identity and attribute providers, 4) a JWT library, and, finally, 5) a local CrySIL node.

Multiple *OpenID Connect endpoints* offer the identity broker's functionality to service providers. Those endpoints follow three parts of the OpenID Connect specification, namely the core specification as well as the extensions for discovery and dynamic registration. Firstly, the discovery extension allows service providers to discover a user's broker and the broker's configuration. Secondly, with dynamic registration, service providers are able to associate with the broker without requiring user interaction. Finally, the implementation of the core specification offers endpoints to authorize the user, exchange authorization code for tokens, and to request further attributes.

Several *databases* store data about users, services, and profiles, as well as temporary data. After a user enrollment process, the user's association with identity and attribute providers along with connection data for her CrySIL node are inserted into the database. As a result of the dynamic registration of a service provider, the registration data of that service is also stored. In addition, the user's decisions made when first associating with a previously unused service are stored as a profile. Such a profile contains choices regarding the user's preferred identity provider, her selection of attribute sources, and the generated re-encryption key. Finally, also some temporary data has to be kept, which includes the issued access tokens with their associated authorization extent.

The broker integrates different *identity and attribute provider plugins*. In order to authenticate a user, the broker delegates this task to an identity provider. Additionally, the broker requires access to a user's encrypted attributes, which are stored on attribute providers. As the particular process steps to achieve these tasks differ from protocol to protocol, the integration of different identity and attribute providers is implemented through exchangeable plugins. The broker implementation includes plugins for the Google Identity Platform as well as for Google Drive.

In order to handle JSON Web Tokens, the broker integrates a *JWT library*. This library is mainly used to verify, issue and re-encrypt tokens. Firstly, tokens obtained during an authentication process, for example from an identity provider that supports OpenID Connect, have to be verified to ensure the user's authenticity. Secondly, the broker issues tokens when the service provider exchanges its authorization code or requests further attributes. Finally, the user's encrypted attributes are tokens, which have to be re-encrypted in order to share them with a service provider.

The implemented broker also includes a *local CrySIL node*, which provides cryptographic operations related to proxy re-encryption. This node is able to re-encrypt the user's attributes locally through a *proxy re-encryption library*. However, this re-encryption operation requires a re-encryption key to transform ciphertexts for the user into ciphertexts for a service provider. Such a re-encryption key is generated at a remote CrySIL node that is trusted by the user, as the generation process operates on the user's private key. In order to acquire permission to involve private key material, the local CrySIL node sends further data, such as tokens and certificate chains, to pass authentication challenges.

In conclusion, service providers delegate user authentication and attribute acquisition to the broker through multiple OpenID Connect endpoints. After discovery of the user's preferred broker and dynamic registration of the service provider, the delegated tasks are performed. Firstly, plugins for multiple identity and attribute providers authenticate the user and acquire authorization to access her attributes. Then, a JWT library initiates the re-encryption of the user's encrypted attributes. This re-encryption is performed by a local CrySIL node, which cooperates with a remote, user-selected CrySIL node to generate re-encryption keys from the user's private key material. Finally, the broker issues an id-token containing the user's re-encrypted attributes, as well as an access token that can be used by the service provider for further requests.

6.3.3. User's CrySIL Node

The CrySIL node is trusted by the user and, therefore, able to provide proxy re-encryption operations that involve the user's private key material. Figure 6.6 illustrates the subcomponents of the implemented CrySIL node. The following paragraphs describe 1) a HTTP receiver, 2) a router that forwards requests to the appropriate actor, 3) the re-encryption actor that operates on the user's key material, and, finally, 4) a gatekeeper that enforces policies.

The *HTTP receiver* acts as the external interface for the CrySIL node. This receiver accepts requests for cryptographic operations from the identity broker and forwards them to the router.

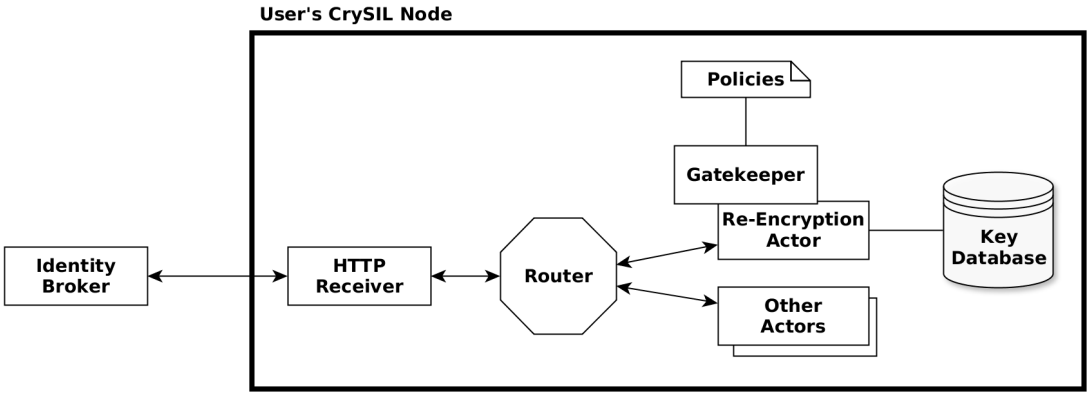


Figure 6.6.: This diagram illustrates the subcomponents of the user's CrySIL node. A HTTP receiver accepts requests for cryptographic operations from the identity broker. These requests are forwarded by the router to a qualified actor. The re-encryption actor handles requests for proxy re-encryptoin operations, such as the generation of a re-encryption key. As private keys may be involved, a gatekeeper governs access to the key material by evaluating user-defined policies.

The *router* passes cryptographic requests onward to an appropriate actor. This routing decision is based on the name of the requested operation as well as the involved key material.

Among other actors, the *re-encryption actor* offers operations related to proxy re-encryption. The primary operation of interest is the generation of re-encryption keys, as this process involves the user's private key material. Especially the involvement and storage of this private key material motivates the use of an external CrySIL node that enjoys the complete trust of the user.

The *gatekeeper* enforces user-specified *policies* that define authentication and authorization requirements, which govern access to the user's private key material. As the broker, an external entity, tries to access the user's private key material in the re-encryption key generation, authentication and authorization processes have to be successfully performed. Users individually define policy files, and, therefore, are able to adjust these policies to the deployment environment of the CrySIL node and other trust requirements. For example, in case the node is located on the user's phone, the policy might require the user's consent by tapping a button to grant access to key material. In the cloud-based implementation of the CrySIL node, users should at least define the following two requirements. Firstly, the broker has to provide a recently issued token that was obtained during the authentication process at an identity provider. This token represents the user's consent. Secondly, the broker has to submit a public key that belongs to a valid service provider as certified by a certification authority. By verifying the certificate chain of the provided public key, the delegation of decryption rights is limited to valid service providers, as opposed to the collusion of a broker and another entity.

In conclusion, the user's CrySIL node offers cryptographic operations, particularly the generation of re-encryption keys, to the identity broker. Requests for these cryptographic operations are accepted by a HTTP receiver and passed through the router to an appropriate actor. The re-encryption actor handles requests regarding proxy re-encryption operations. In particular, the re-encryption key generation is of primary interest. As this key generation involves the user's private key material, a gatekeeper enforces user-defined policies regarding authentication

and authorization.

6.4. Chapter Conclusion

In this chapter, the implementation of the proposed identity broker system has been described. This broker system allows users to share their attributes in a controlled fashion by using proxy re-encryption. The required re-encryption key is generated on a device that enjoys the user's complete trust. The implementation consists of multiple components that interact with each other in order to authenticate the user and acquire her attributes. This process was designed to require a minimum amount of user interaction to provide a pleasant experience.

The user authentication and attribute acquisition process involves multiple components, namely a service provider, the broker, identity and attribute providers, as well as the user's CrySIL node. After a setup phase, these components perform the following steps: 1) The service provider delegates authentication and attribute acquisition to the identity broker, which 2) builds a profile about the user, 3) authenticates the user at an identity provider, and 4) obtains authorization to access the user's attribute provider. 5) Then, the broker retrieves the user's encrypted attributes, generates a re-encryption key via the user's CrySIL node, and re-encrypts the attributes for the service provider. 6) The service provider has to exchange its authentication code for tokens, 7) which can be used to request further attributes. 8) Finally, the service provider is able to fulfill the user's initial request.

In order to provide a pleasant user experience, the user is only involved in a subset of the before mentioned process steps. 1) The user initiates the process by requesting a protected resource at the service provider. 2) This service determines the username to discover and if necessary register at the user's preferred identity broker. 3) Then, the user builds a profile by selecting identity and attribute providers. 4) Authentication and 5) authorization has to be performed at these providers. 6) Subsequently, the user selects attributes to share. 7) After granting final consent, 8) the service provider receives all necessary information to fulfill the user's initial request. By reusing stored information in subsequent runs, it is possible to cut down the number of required interactions. Such information includes decisions stored in a profile on the broker or authentication information in a SSO session.

7. Security Analysis

This chapter presents a security analysis of the implemented identity broker system. The analysis roughly follows the Common Criteria for Information Technology Security Evaluation, which is standardized in ISO/IEC 15408 [ISO09]. An evaluation with Common Criteria consists of two main parts. First, a generic description of the evaluated system is provided. This description includes the target of evaluation, involved actors, security assumptions, valuable assets, threat agents, threats and security objectives. Then, this basis is used to evaluate the security of an implemented system. In order to protect against threats, such a system has to implement countermeasures that fulfill the security objectives.

This chapter is organized as follows: Section 7.1 describes the target of evaluation and the involved actors. Then, section 7.2 states security assumptions made for the analysis. In section 7.3, assets valuable to the actors are specified. Subsequently, section 7.4 defines agents that pose a threat to the assets, due to their capabilities and motivation. Section 7.5 elaborates on the individual threats. In section 7.6, generic security objectives describe how to avert these threats. Then, section 7.7 explains how these objectives were achieved in the implementation of the identity broker system. Finally, section 7.8 concludes the security analysis and this chapter.

7.1. Target of Evaluation and Actors

As the first step, the scope of this security evaluation is defined by specifying the target of evaluation (TOE) as well as the involved actors. The target of evaluation consists of actors or their components that were developed or extended in this thesis. During the design phase of these components, countermeasures were planned to achieve security objectives which avert threats to the systems assets. However, there are further actors that reside outside the scope of this security analysis. In particular, the secure communication with these actors is of interest in the analysis. The following lines define the individual involved actors as well as their scope regarding this analysis. Figure 7.1 illustrates the resulting target of evaluation.

User: The user plays a major role, as she initiates the authentication and attribute acquisition use case. During this use case, the user interacts with multiple other actors. The involved communication with those actors is of interest to the analysis.

Service Provider: The service provider offers a service, which includes resources that require protection. This protection is implemented by a protection library, which delegates both the authentication of users as well as the attribute acquisition to an identity broker. In the security analysis, this protection library is part of the target of evaluation.

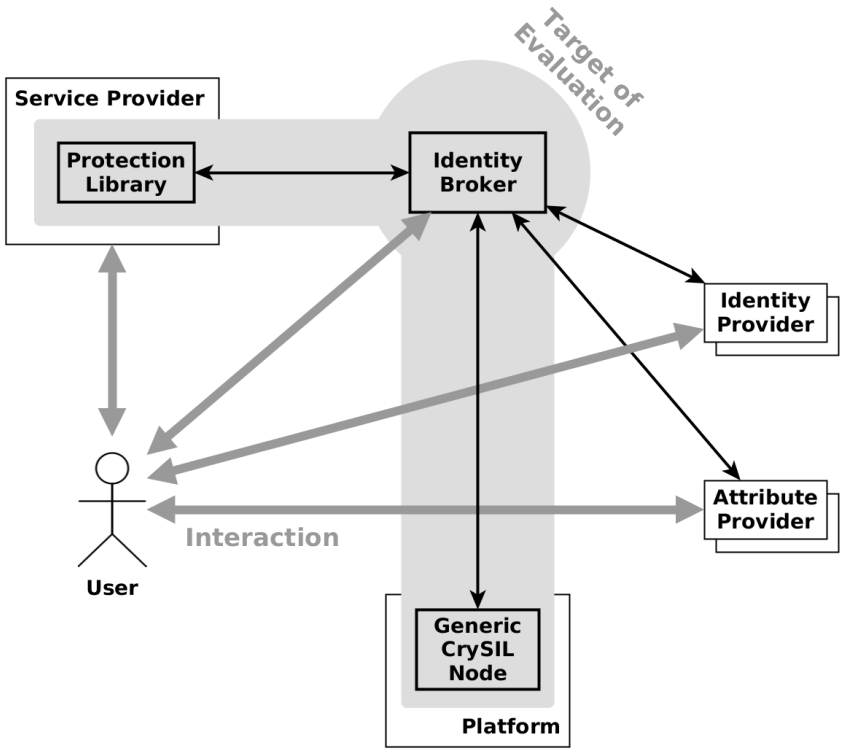


Figure 7.1.: This figure illustrates the target of evaluation for the security analysis based on the involved actors and their components. The target of evaluation is mainly concerned with components developed or extended in this thesis. Those components include an identity broker, a protection library for service providers, and an extended CrySIL node. However, communication with other actors, namely users as well as identity and attribute providers, also have to be considered.

Identity Broker: The identity broker represents the heart of this thesis. It offers authentication via identity providers as well as re-encryption of attributes obtained from attribute providers. As this broker may be operated in the cloud environment, the associated threats had to be considered during the design phase. Furthermore, the sharing of sensitive data through proxy re-encryption and the provisioning of the required key material is a major part of the innovative focus of this thesis and therefore also of particular interest in the security analysis. Consequently, the identity broker is a central part of the target of evaluation.

User's CrySIL Node: The user's CrySIL node offers cryptographic operations involving her private key material. In the identity broker system, this node handles the broker's requests to generate re-encryption keys that transform the user's encrypted attributes to ciphertext for a service provider. As proxy re-encryption functionality was introduced into the node in the course of this thesis, those operations and their impact are part of the target of evaluation. However, since CrySIL nodes can be operated in a multitude of environments, the platform-specific implementations are outside the scope of this analysis. As a result, the focus lies on the proxy re-encryption capabilities and the involved access to the user's key material.

Identity Provider: The identity provider authenticates the user for the identity broker. As any number of identity providers can be integrated with the broker via plugins, these identity providers are outside the scope of this security analysis. However, the communication channel between broker and identity provider as well as the transmitted data have to be protected. Therefore, this analysis considers such communication.

Attribute Provider: The attribute provider stores the user's attributes and offers them to authorized identity brokers. As with identity providers, plugins can integrate any attribute provider at a broker. Therefore, the target of evaluation does not include attribute providers, but involved communications are considered in this security analysis.

In conclusion, the target of evaluation limits the scope of the security analysis to central actors and components that were developed or extended in the course of this thesis. The TOE mainly consists of the identity broker, the user's CrySIL node as well as the protection library at a service provider. Additionally, the security analysis considers the communication with other involved actors, namely the user, identity and attribute provider.

7.2. Security Assumptions

The security analysis is based on the security assumption defined in this section. These assumptions can be classified into two groups, namely trust and general assumptions. First, the trust assumptions regarding the individual server-side actors are specified. Subsequently, further assumptions define the initial conditions as well as the general operation of the identity broker system.

Trust assumptions have to be made to later specify the power of adversaries and possible attacks. The following lines explain the trust assumptions of this security analysis for the individual

server-side actors.

AS.1 **User's CrySIL Node is Trustworthy:** The user selects a trustworthy environment to operate her CrySIL Node, which handles private key material.

AS.2 **Identity Provider is Trustworthy:** In this security analysis, the identity provider is considered to be trustworthy.

AS.3 **Identity Broker is Honest But Curious:** As explained in the section about threat agents, an identity broker running in the cloud attracts multiple internal and external attackers, which are interested in the sensitive data processed by the broker. However, if those adversaries get access to the broker, malicious behavior would increase the chances of their discovery. Therefore, it is a reasonable assumption that such a cloud-based identity broker is honest but curious. Furthermore, such an identity provider may perform additional processing and communications to satisfy its curiosity.

AS.4 **Service Provider is Honest But Curious:** Like the identity provider, the service provider is considered to be honest but curious. As this service provider may also be operated in the cloud, the same reasoning applies. However, the service provider is permitted to read an authorized part of the user's sensitive data.

AS.5 **Attribute Provider is Possibly Malicious:** For the attribute provider, no trust assumptions limiting the power of attackers are defined. Therefore, the attribute provider may also be malicious. As a result, malicious behavior has to be detected in the identity broker system.

Furthermore, this security analysis is also based on some general assumptions. These assumptions describe the initial conditions of the identity broker system and assure the intended operation. Below, the general assumptions are explained in further detail.

AS.6 **Correct Implementation:** This security analysis assumes that all software components were implemented correctly and do not have security vulnerabilities.

AS.7 **Correct Installation:** All software is assumed to be correctly installed. The setup process includes the generation of the required key material.

AS.8 **Correct Operation:** It is assumed that all involved parties operate software components under their control correctly. For example, this includes that users and providers do not expose key material due to carelessness.

AS.9 **Valid Certificates:** This assumption states that actors are in possession of relevant valid certificates. For server-side actors, this also includes certificates for TLS. Additionally, service providers need a certificate for their proxy re-encryption public key from an approved authority.

AS.10 **Availability:** Finally, this security analysis assumes all server-side actors are available during the process of user authentication and attribute acquisition.

In conclusion, multiple security assumptions establish the context of the security analysis. These assumptions can be grouped into two classes, namely trust assumptions for the individual

server-side actors as well as general assumptions regarding the initial conditions and operation of the identity broker system.

7.3. Assets

This section defines assets that are valuable to the involved actors and therefore have to be protected by the target of evaluation. The two primary assets are: 1) sensitive data about the user, and 2) resources of the service provider that require authentication and attributes. In order to protect these primary assets, further assets are defined. The following lines describe the individual assets in further detail.

A.1 **Sensitive Data:** The user's attributes are regarded as sensitive data. For example, such attributes could include the user's address, credit card information or medical records. The attributes are stored at the attribute provider and handled by the identity broker. Eventually, a selected subset of those attributes is processed at the service provider.

A.2 **Encrypted Sensitive Data:** Ciphertexts containing sensitive data are also considered being an asset in this security analysis. In order to minimize the impact of security vulnerabilities, security measures have to be employed on multiple levels. Therefore, by protecting direct encryptions of sensitive data, the amount of available ciphertext is limited even in case an attacker obtains decryption key material.

A.3 **Key Material:** Private cryptographic key material protects both the confidentiality of the contents of encrypted data as well as the integrity of signed data. Therefore, in order to ensure the confidentiality and integrity of data, such private key material has to be protected. In the identity broker system, the user's CrySIL node is of particular interest, as it holds the user's private encryption key material.

A.4 **Protected Resources at SP:** The service provider has to protect some of its resources by requiring authentication and a subset of the user's attributes. For example, only an authenticated and authorized user should be able to order goods or delete her account. Motivated by this need, the service provider relies on the identity broker system and its local protection library.

A.5 **Protected Resources at IdP, AP and Broker:** Identity and attribute providers as well as identity brokers also offer resources that require protection. For example, access to the user's attributes stored at the attribute provider should only be available to authorized entities.

A.6 **Id and Access Tokens:** As result of a successful authentication and authorization process, id and access tokens are yielded. These tokens are widely used in the identity broker system to represent the acquired authorization and therefore can be used to gain access to resources. In consequence, such tokens have to be protected.

A.7 **Session Data:** Similar to the tokens, a HTTP session can contain the outcome of a authentication and authorization process. However, while a token can be used between any

actors, a session is established between the user's browser and a web server. These servers represent the service, identity and attribute providers as well as identity brokers.

In conclusion, then identity broker system handles multiple assets that are of value to the involved actors. Primarily, the user's sensitive data and a selection of the service provider's resources have to be protected. The protection of these assets also involves further assets. In the following sections, threats to the defined assets are examined and subsequently objectives and countermeasures to avert those threats are described.

7.4. Threat Agents

This section describes agents that present a threat to the previously defined assets due to their capabilities and motivation. A focus is put on attackers introduced by the deployment of components in the cloud. However, traditional threat agents that are capable to make requests or alter traffic are not neglected. The following lines provide a detailed description of the individual attackers along with their motivations and capabilities.

TA.1 **Cloud Attacker:** Even though the cloud as deployment location offers remarkable advantages, such an environment also introduces new risks. Attackers who gain access to the cloud provider acquire control of the hosted services. Such attackers include: 1) external attackers that penetrate the cloud provider's defenses, 2) internal attackers, such as employees of the cloud provider that already have access, and 3) law enforcement agencies of countries where the computing resources of the cloud provider are located. In the following, attackers on the individual services that might be hosted by cloud providers are defined with regards to the security assumptions.

TA.1.1 **Identity Broker Attacker:** The identity broker and its deployment environment are generally considered to be honest but curious. That is, the broker performs its operations correctly, but is curious and therefore inspects the information passing through. Furthermore, the broker is also allowed to make additional requests and computations to satisfy its curiosity, as long as they do not have an impact on the broker's usual operation. As such an identity broker in the cloud might be interested in compromising the confidentiality of assets, it can also be considered as an attacker.

TA.1.2 **Service Provider Attacker:** Services might also be deployed to the cloud. According to the previously defined security assumptions, the service provider is also honest but curious with extended capabilities, analogous to the identity broker. In contrast to the broker, the service provider is allowed to see a portion of the user's sensitive data.

TA.1.3 **Attribute Provider Attacker:** As defined in the security assumptions, attribute providers could possibly be malicious. That is, these providers actively deviate from their legitimate operation, for example by modifying or exchanging attributes.

TA.2 **Web Attacker:** A web attacker is able to send and receive arbitrary packets, such as HTTP requests and responses. Furthermore, this type of attacker controls multiple domains

to host resources, for example malicious Javascript code. These capabilities are widely available to internet users. Web attackers are motivated to collect information about the user or to gain access to protected resources without the required privileges.

TA.3 **Network Attacker:** Network attackers are capable to mount man-in-the-middle attacks. In such an attack, the attackers are able to inspect and modify the traffic between all entities on the network. For example, an internal or external agent that compromised an internet service provider (ISP) holds these capabilities. Like in the case of a web attacker, a network attacker strives to learn sensitive data about the user and to obtain access to protected resources.

In conclusion, the identity broker system has to deal with multiple threat agents. Some of these agents, namely web and network attackers, are only able to interact with the system from the outside through communication channels. However, cloud attackers are more powerful, as they are capable to observe or even influence the internal operation of system components. In the following section, these different types of agents are used as a basis to model the individual threats.

7.5. Threats

The threat agents defined in the previous section have both the capability and motivation to pose multiple threats to the identity broker system's assets. In the following lines, the individual threats, affected assets, and involved threat agents are described in further detail. Table 7.1 presents a summary of the threats.

T.1 **SP Acquires Non-Minimial Data Set:** As result of a legitimate attribute acquisition process, the service provider might not only receive the requested data, but also additional information that is coupled with the required attributes. For example, if the service provider requests the user's name, the system would violate the user's privacy by disclosing the user's whole digital driver's license even though it contains the name. A service provider might exploit such non-minimal data set to learn further sensitive information about the user.

Affected Assets: A.1 Sensitive Data
Involved Threat Agents: TA.1.2 Service Provider Attacker

T.2 **SP Re-Uses Outdated Permissions:** In order to improve the usability of the identity broker system, components store the user's authentication information and authorization decisions to avoid repetitive interaction. However, by reusing these decisions after some amount of time, they might not reflect the user's current intentions anymore. The validity time of tokens and session data for identity and attribute providers are out of scope. Instead, the focus of this threat is the authorization process at the identity broker. If this broker reuses the user's decisions for an indefinite amount of time, service providers are able to reacquire tokens regardless of the user's actual intentions. Those tokens include fresh sensitive data about the user and can be used as proof of authorization.

Affected Assets: A.1 Sensitive Data, A.6 Id and Access Tokens, A.7 Session Data
Involved Threat Agents: TA.1.2 Service Provider Attacker

T.3 **Broker or Web Attacker Utilize User's Keys for Unintended Operations:** Attackers might misuse the access to the user's key material, which has to be provided to generate re-encryption keys, for unintended purposes. The user's CrySIL node stores her key material and accesses it to fulfill cryptographic requests. This key material has to be indirectly used in an operation triggered by the broker in order to generate a re-encryption key. However, the broker and other attackers could exploit the ability to perform arbitrary operations with the user's keys, for example by decrypting the user's attributes.

Affected Assets: A.3 Key Material
Involved Threat Agents: TA.1.1 Identity Broker Attacker, TA.2 Web Attacker

T.4 **Broker Generates a Re-Encryption Key for Itself:** Assuming the CrySIL node only allows the broker to generate re-encryption keys using the user's private key, the broker could still exploit these permissions. The broker requires a re-encryption key to transform encrypted attributes for the user to ciphertext for a service provider. These re-encryption keys are generated by the CrySIL node from the stored user's private key and the service provider's public key. However, since the broker supplies the service provider's key material, the broker could present the public key of a known key pair. The resulting re-encryption key would enable the broker to re-encrypt the user's attributes containing sensitive data for itself. In consequence, with the private key of the known key pair the broker would be able to decrypt the attributes.

Affected Assets: A.1 Sensitive Data, A.3 Key Material
Involved Threat Agents: TA.1.1 Identity Broker Attacker

T.5 **Broker or AP Read Sensitive Data:** In the identity broker system, multiple software components running in the cloud handle the user's attributes. Attribute providers store the user's data. Identity brokers access the attribute providers to compile a set of attributes requested by the service providers. As previously described, the cloud environment introduces attackers that have both the capability and motivation to inspect the processed data. Therefore, attackers of identity brokers and attribute providers would learn data involving sensitive information. However, service provider attackers are not considered in this threat, as they are only given a set of attributes, which the service provider is allowed to decrypt anyway.

Affected Assets: A.1 Sensitive Data
Involved Threat Agents: TA.1.1 Identity Broker Attacker, TA.1.3 Attribute Provider Attacker

T.6 **AP Modifies Sensitive Data:** As the attribute provider is operated in the cloud, an attacker might be able to modify the attributes a user stored there if those attributes are not integrity-protected. Consequently, this might pose a problem for the user who supposedly supplied the data or a service provider that relies on the obtained information. Due to the security assumptions, neither identity brokers nor service providers modify the processed data.

Affected Assets: A.1 Sensitive Data
Involved Threat Agents: TA.1.3 Attribute Provider Attacker

T.7 **AP Offers Exchanged Sensitive Data:** Assuming the integrity of the attributes is protected, a malicious attribute provider might still be able to offer information that was not supplied by the involved user. For example, for the request of an identity broker, the attribute provider attacker could reply with a valid attribute of another user. In consequence, if the authenticity of the attributes is not ensured, these attributes could be exchanged. Again, identity brokers and service providers are not considered in this threat due to the security assumptions.

Affected Assets: A.1 Sensitive Data
Involved Threat Agents: TA.1.3 Attribute Provider Attacker

T.8 **Network Attacker Accesses Data During Transport:** As network attackers are able to read and modify all network traffic between the actors of the identity broker system, this capability poses a threat to the transmitted assets. The involved attributes are manifold: A subset of the user's attributes containing sensitive data is requested by the identity broker from the attribute provider and then delivered to the service provider. Re-encryption keys generated at the user's CrySIL node are sent to the broker. Id and access tokens are issued to authorized parties to protect the access to APIs on identity broker, attribute provider and service provider. Lastly, sessions are established between the user and multiple parties. By obtaining such assets, the attacker learns sensitive information about the user, is able to compromise authenticity mechanisms or can peel off protective layers.

Affected Assets: A.1 Sensitive Data, A.2 Encrypted Sensitive Data, A.3 Key Material, A.6 Id and Access Tokens, A.7 Session Data
Involved Threat Agents: TA.3 Network Attacker

T.9 **Web Attacker Injects Code:** A web attacker might use different attack vectors to execute code in the context of the user's browser. By embedding data from untrusted sources directly into the document object model (DOM), a page is susceptible to cross site scripting (CSS) attacks. In such an attack, the input data is interpreted as code and therefore executed. Additionally, including external resources into the page, such as Javascript libraries or iframes, also poses a risk, as the provider of these resources is able to inject arbitrary code. As a result, the attacker is able to obtain all data that is shared with the page. This includes sensitive information that is displayed as well as tokens and session data exposed to the browser.

Affected Assets: A.1 Sensitive Data, A.6 Id and Access Tokens, A.7 Session Data
Involved Threat Agents: TA.2 Web Attacker

T.10 **Web Attacker Performs SQL Injection:** SQL injection might allow an attacker to obtain and manipulate data, as well as to change the process flow of the software. A service might be susceptible to a SQL injection if input data provided by a web attacker is directly inserted into a database query without consideration for special characters. Consequently, this allows attackers to modify the query and thereby induce unintended behavior. A successful SQL injection has two ramifications. Firstly, all information stored in the database or directly involved in the queries is vulnerable to disclosure or manipulation.

This data includes but is not limited to sensitive possibly encrypted information, tokens and sessions. Secondly, process flow of the service can be influenced, as it depends on the outcome of the database queries. For example, by omitting authentication conditions in the query, an attacker might be considered to be authenticated and hence receive tokens and session data.

Affected Assets: A.1 Sensitive Data, A.2 Encrypted Sensitive Data, A.6 Id and Access Tokens, A.7 Session Data
Involved Threat Agents: TA.2 Web Attacker

T.11 **Web Attacker Makes Unauthorized Use of Protected Resource:** With cross-site request forgery (CSRF), a web attacker could reuse sessions established between the user and an honest web application to access protected resources at that application. By tricking a user to request a resource at the honest application, her browser might include a previously issued session cookie. From the application's perspective, the request would be perceived as legitimate and made by an authorized user. Even though the capability to read information of such an attack is limited, adversaries might still be able to exploit the caused side effects of such requests.

Affected Assets: A.4 Protected Resources at SP, A.5 Protected Resources at IdP, AP and Broker, A.7 Session Data
Involved Threat Agents: TA.2 Web Attacker

In conclusion, the proposed identity broker system faces multiple threats. These threats are caused by previously described threat agents, who have the capabilities and motivation to attack the system's assets. Motivated by these threats, the next two sections first present security objectives and then introduce countermeasures.

7.6. Security Objectives

In order to avert the threats presented in the previous section, the TOE has to satisfy multiple security objectives. While these objectives provide a general description, the next section goes into detail about their implementation. The following lines list the individual security objectives and covered threats. Table 7.2 illustrates a mapping betwen objectives and threats.

O.1 **Fine Granularity of Attributes:** The user's sensitive data is stored separately in attributes with fine granularity at the attribute provider. This ensures that the identity broker is able to fulfill a service provider's request by compiling a minimal set of attributes and thereby only discloses the necessary information.

Covered Threats: T.1 SP Acquires Non-Minimial Data Set

O.2 **Explicit User Consent:** The TOE ensures that the user explicitly grants consent every time a service provider requests a short-lived proof of authorization. Consequently, service providers are not able to reuse outdated authorization decisions, which do not reflect the user's current intentions.

Covered Threats: T.2 SP Re-Uses Outdated Permissions

O.3 **Protect Access to CrySIL Operations:** Access to both the operations provided by the user's CrySIL node as well as the involved key material is protected by policies. These policies limit the broker to only use a legitimate combination of operations and key material .

Covered Threats: T.3 Broker or Web Attacker Utilize User's Keys for Unintended Operations

O.4 **Ensure that Broker Does Not Re-Encrypt for Itself:** The policies governing the access to the user's CrySIL node also prevent identity brokers from successfully requesting a re-encryption key for a bogus service provider. Consequently, identity brokers are not able to obtain a re-encryption key for a fake service provider's public key, which belongs to a known key pair. As a result, brokers are not able to re-encrypt the user's attributes and then decrypt the sensitive data with the known private key.

Covered Threats: T.4 Broker Generates a Re-Encryption Key for Itself

O.5 **Protect Confidentiality of Sensitive Data:** The TOE ensures the confidentiality of the user's sensitive data, which make up her attributes. Therefore, neither attribute providers, that store the user's attributes, nor brokers, who handle the attributes, are able to learn any contained sensitive information.

Covered Threats: T.5 Broker or AP Read Sensitive Data

O.6 **Protect Integrity and Authenticity of Sensitive Data:** The TOE also ensures both the integrity and authenticity of the user's sensitive data. Firstly, integrity-protection prevents attribute providers from modifying sensitive data. Secondly, the attribute's authenticity of origin ensures that the user's attribute was not exchanged with data of another user.

Covered Threats: T.6 AP Modifies Sensitive Data, T.7 AP Offers Exchanged Sensitive Data

O.7 **Employ State-of-the-Art Transport Security:** The TOE protects all network traffic between the identity broker system's components with state-of-the-art transport layer security. Consequently, network attackers are not able to inspect or modify data transmitted through secure tunnels.

Covered Threats: T.8 Network Attacker Accesses Data During Transport

O.8 **Validate and Escape Untrusted Input:** The TOE treats all input from untrusted sources as possibly hostile. Therefore, that data is validated and escaped according to the used environment. As a result, attackers are not able to inject code into web pages or modify database queries.

Covered Threats: T.9 Web Attacker Injects Code, T.10 Web Attacker Performs SQL Injection

O.9 **Do Not Include Untrusted Code:** The TOE does not embed source code from untrusted sources. Therefore, attackers are not able to inject malicious code by compromising such an untrusted source.

Covered Threats: T.9 Web Attacker Injects Code

O.10 **Protection of Server-Side APIs:** The TOE protects server-side APIs against unauthorized

and unintended use. Therefore, attackers are not able to access protected resources, for example by tricking a user to make a request, which reuses an established session.

Covered Threats: T.11 Web Attacker Makes Unauthorized Use of Protected Resource

In conclusion, to avert the threats presented in the previous section, the TOE has to satisfy multiple security objectives. These security objectives are implemented as countermeasures, which are described in the following section.

7.7. Countermeasures

This section explains the countermeasures taken to avert the previously described threats. In order to protect the system's assets from threats, security objectives were defined. The implementation of the proposed identity broker system realizes these objectives with specific countermeasures. Below, the countermeasures implementing the individual security objectives are explained in detail.

O.1 **Fine Granularity of Attributes:** In order to only disclose required information, the user's attributes have to be available in fine granularity. The user controls the granularity, as she divides her data into individual JSON web tokens before uploading them to the attribute provider. Such a token contains a single data item or highly related items that have to be disclosed together. A list of these tokens containing all the required information is compiled by the broker to satisfy a service provider's request. However, due to the fine granularity of the attribute tokens, no additional data has to be revealed.

O.2 **Explicit User Consent:** Explicit consent from the user is required during the authentication and authorization process to ensure that the user's previous decisions still reflect her current intentions. Otherwise, service providers would be able to acquire fresh attributes containing sensitive data about the user or proofs of authentication. As presented in section 6.2, the broker presents a concise overview of both the service provider's request as well as previously made decisions to the user. This overview shows the attributes requested by the service provider, the previously selected attribute providers as origins for these attributes, and the earlier chosen identity provider. As a result, the user has the opportunity to apply modifications or consent with a single click in case the stored decisions are still aligned with her current intentions.

O.3 **Protect Access to CrySIL Operations:** As the user's CrySIL node provides operations that access her private key, those operations have to be protected. Therefore, once an operation is requested the CrySIL node has to reach an access decision before actually involving the user's key material. This access control is performed by an authorization module, which evaluates eXtensible Access Control Markup Language (XACML) policies specified by the user. These versatile policies may require further data about the requester, such as a demonstration of authentication. Hence, the requester has to present the required data through additional communication.

In the identity broker system, the policies not only have to protect the access to the user's node, but also have to comply with the usability objective of minimizing user

interaction. These security and usability requirements are implemented by a policy stating that brokers may only request the re-encryption key generation operation involving an appropriate private key after a recent id-token issued by a trusted identity provider for the user was supplied. This id-token attests two facts: Firstly, the user recently performed an authentication and consequently came into possession of the token. Secondly, as the broker supplies the token, the user trusts the broker to some degree and therefore deliberately shared her token. As a result, only entities who enjoy enough trust to get access to the user's id-token may order the generation of a re-encryption key involving the user's private key.

O.4 **Ensure that Broker Does Not Re-Encrypt for Itself:** Additional policies have to be introduced to prevent brokers from obtaining a re-encryption key that is able to translate the user's attributes to ciphertext decryptable with a known private key. Brokers supply the service provider's public key for the re-encryption key generation, which takes place at the user's CrySIL node. This CrySIL node has to ensure that the supplied public key belongs to a valid service provider. In the identity broker system, the link between public key and service provider is certified by a certification authority. Therefore, the user's CrySIL node includes a policy which requires the incoming public key to be certified by this certification authority. Assuming this certification authority is trustworthy, valid certificates are not issued to brokers for their key material. Consequently, these brokers are not able to obtain a re-encryption key, which allows them to first re-encryt and subsequently decrypt the user's encrypted attributes.

O.5 **Protect Confidentiality of Sensitive Data:** In order to protect the confidentiality of sensitive data stored and handled in inquisitive environments, these data are encrypted with proxy re-encryption. The user encrypts her attributes with such a proxy re-encryption scheme before uploading them to an attribute provider. This attribute provider is not able to inspect the contents of these attribute files. However, convenient sharing with other recipients, such as service providers, is still possible with proxy re-encryption. To provide data to a service provider, the identity broker acts as re-encryption proxy and translates the attributes encrypted for the user to ciphertext for the service provider. One integral feature of proxy re-encryption states that the proxy does not learn the underlying plaintext at any stage during the re-encryption process. However, as discussed in section 4.3, the selected scheme also has to fulfill two properties, namely being unidirectional and CPA-secure. Firstly, if the scheme is not unidirectional, the broker can use a clever combination of re-encryption keys to re-encrypt the user's attributes for a service provider which the user never authorized. Secondly, if the scheme is not CPA-secure, the broker could encrypt all candidates of a small message space and compare the resulting ciphertexts to the user's encrypted attributes. However, this second point is typically not an issue in the implementation with JWT, as these tokens include additional data in the encrypted payload and therefore have a larger message space. As a result, proxy re-encryption allows to preserve the confidentiality of sensitive data when stored and shared in untrusted environments.

O.6 **Protect Integrity and Authenticity of Sensitive Data:** The integrity and authenticity of a user's sensitive data is protected by a digital signature. This signature can either be self-signed by the user or issued by another entity that verified the attribute. The user's attributes are signed before they are uploaded to the attribute provider. By verifying the

signature, the service provider examines the integrity and authenticity of the requested attributes. Firstly, the integrity is ensured as modifications to the signed content invalidate the signature. Therefore, an attacker is not able to successfully alter the user's attributes. Secondly, the authenticity of origin is guaranteed, since a signature created with the user's private key can only be verified with the corresponding public key. Hence, even if a requested attribute of one user is exchanged with a signed attribute of another user, the service provider still realizes that the expected public key fails during the verification. As a result, the integrity and authenticity of the user's attributes are protected.

O.7 **Employ State-of-the-Art Transport Security:** In order to prevent attackers from inspecting or modifying network traffic, all communications between actors in the identity broker system are protected by transport layer security (TLS). When the system is set up, the web servers acquire valid certificates from a trusted certificate authority. To establish a secure connection, these certificates have to be sent to the requester. It is the responsibility of the requester to verify the received certificate. With a certificate the server authenticates to a requester and therefore cannot be impersonated. In addition, certificates allow the communication partners to negotiate session keys, which are used to encrypt the following traffic. As a result, data in transport is protected from attackers wanting to inspect or modify it.

O.8 **Validate and Escape Untrusted Input:** Input data from untrusted sources has to be handled correctly, namely validated and escaped, in order to eliminate unintended influence on the process flow. Validation has to be performed to meet the requirements and expectations of the procedure that processes the input data. For example, standards defining a protocol may also point out important validation considerations. The method for escaping depends on the environment that uses the input data. Prominent target environments for attacks based on not-escaped data are web pages and database queries. In the web environment, the used templating engine escapes the input data before inserting it into a template and subsequently into the page. Therefore, untrusted data cannot mistakenly be evaluated as HTML markup or Javascript code. Additionally, input data is not directly embedded into database queries. Instead, prepared statements are used, which accept input data as parameters that are automatically escaped. Consequently, the structure of the queries cannot be altered by an attacker. In conclusion, input data has to be both validated according to the procedure's expectations and escaped to prevent injection of code in the used environment.

O.9 **Do Not Include Untrusted Code:** The web pages served by components of the identity broker system do not include external resources, as this represents another attack vector. An attacker could compromise a provider of these resources and therefore this attacker would also be able to inject content into the served page. In case the included resource is a javascript library, the adversary gains full access to the page's context. With an iframe, the attacker would be limited by the same origin policy. However, by hosting additional resources required by the page directly at the same server, adversaries are not able to exploit this attack vector.

O.10 **Protection of Server-Side APIs:** In the identity broker system, the endpoints are protected against unauthorized and unintended use by a series of mechanisms. The protected

resources require a proof of the requester's authorization, either through supplied session data, an id-token or an access-token. The supplied information is then verified by the endpoint. Also, as described in O.8, the sent input data is validated against the endpoint's expectations. However, these precautions do not protect against CSRF attacks. In such an attack, requests are issued by the user's browser while visiting another, possibly compromised, domain. These requests might also carry previously established session-data which authorize the user. There are a number of ways to avert this problem. The `Referer` and `Origin` headers allow to discard requests from unexpected origins. Alternatively, the server may require the request to also carry a previously shared token. As a result, unauthorized as well as unintended requests for protected resources are discarded.

In conclusion, multiple countermeasures were implemented to fulfill the previously stated security objectives. While the security objectives described generic goals, the countermeasures explained how these goals were reached by the implementation of the identity broker system.

7.8. Chapter Conclusion

In this chapter, a security analysis of the implemented identity broker system was performed. The methodology derived from the Common Criteria for Information Technology Security Evaluation consisted of the following seven steps. 1) First, the target of evaluation and the involved actors were defined. This limited the scope of the analysis to components that were developed or extended in the course of this thesis and their communications. 2) Then, security assumptions were stated. These assumptions were either trust assumptions for the individual actors or general assumptions regarding the initial conditions and operation of the system. 3) In addition, assets that are valuable to the actors were specified. Primarily, these assets are the user's sensitive data and protected resources of the service provider. 4) Subsequently, agents that pose a threat to these assets were defined. The security analysis placed a focus on attackers that gain access to the cloud services, but traditional adversaries who participate in the network or have the ability to influence the traffic were also examined. 5) Also, details on the individual threats were given. These threats were made possible by the threat agent's capabilities 6) Then, generic security objectives to avert these threats were described. 7) Finally, an explanation was provided how the identity broker system fulfills these security objectives by implementing countermeasures against threats.

	Assets							Threat Agents				
	A.1 Sensitive Data	A.2 Encrypted Sensitive Data	A.3 Key Material	A.4 Protected Resources at SP	A.5 Protected Resources at IdP, AP and Broker	A.6 Id and Access Tokens	A.7 Session Data	TA.1.1 Identity Broker Attacker	TA.1.2 Service Provider Attacker	TA.1.3 Attribute Provider Attacker	TA.2 Web Attacker	TA.3 Network Attacker
T.1 SP Acquires Non-Minimial Data Set	✓								✓			
T.2 SP Re-Uses Outdated Permissions	✓					✓	✓		✓			
T.3 Broker or Web Attacker Utilize User's Keys for Unintended Operations			✓					✓			✓	
T.4 Broker Generates a Re-Encryption Key for Itself	✓							✓				
T.5 Broker or AP Read Sensitive Data	✓							✓		✓		
T.6 AP Modifies Sensitive Data	✓									✓		
T.7 AP Offers Exchanged Sensitive Data	✓									✓		
T.8 Network Attacker Accesses Data During Transport	✓	✓	✓			✓	✓					✓
T.9 Web Attacker Injects Code	✓					✓	✓				✓	
T.10 Web Attacker Performs SQL Injection	✓	✓				✓	✓				✓	
T.11 Web Attacker Makes Unauthorized Use of Protected Resource				✓	✓		✓				✓	

Table 7.1.: This table gives an overview of threats to assets as well as the corresponding threat agents.

	T.1 SP Acquires Non-Minimial Data Set	T.2 SP Re-Uses Outdated Permissions	T.3 Broker or Web Attacker Utilize User's Keys for Unintended Operations	T.4 Broker Generates a Re-Encryption Key for Itself	T.5 Broker or AP Read Sensitive Data	T.6 AP Modifies Sensitive Data	T.7 AP Offers Exchanged Sensitive Data	T.8 Network Attacker Accesses Data During Transport	T.9 Web Attacker Injects Code	T.10 Web Attacker Performs SQL Injection	T.11 Web Attacker Makes Unauthorized Use of Protected Resource
O.1 Fine Granularity of Attributes	✓										
O.2 Explicit User Consent		✓									
O.3 Protect Access to CrySIL Operations			✓								
O.4 Ensure that Broker Does Not Re-Encrypt for Itself				✓							
O.5 Protect Confidentiality of Sensitive Data					✓						
O.6 Protect Integrity and Authenticity of Sensitive Data						✓	✓				
O.7 Employ State-of-the-Art Transport Security								✓			
O.8 Validate and Escape Untrusted Input									✓	✓	
O.9 Do Not Include Untrusted Code									✓		
O.10 Protection of Server-Side APIs											✓

Table 7.2.: This table illustrates which threats are covered by the individual security objectives.

8. Future Work

For future work, multiple research directions seem promising to yield improvements to the proposed identity broker system. Especially, the introduction of innovative cryptographic primitives as well as the extension of already used cryptography could be beneficial. The following lines give a short overview of encouraging research areas.

Conditional Proxy Re-Encryption: By replacing traditional proxy re-encryption with conditional proxy re-encryption [Wen+09] and especially fine-grained conditional proxy re-encryption [Fan+11], the user could significantly limit the re-encryption power of a malicious broker. With conditional proxy re-encryption, the users attach conditions to the ciphertexts. Such a ciphertext can only be translated for another recipient if the supplied re-encryption key satisfies the applied conditions. For example, users could tag their `medical` data and only allow the broker to re-encrypt such `medical` ciphertext for a specific hospital as service provider, by supplying an appropriate re-encryption key. As a result, a malicious broker would be limited by the re-encryption key's ability to only re-encrypt certain ciphertexts.

Redactable Signatures: By integrating redactable signatures [Joh+02] with proxy re-encryption, an external issuer could provide a signed document, while only the minimal required parts of that document would have to be revealed to a service provider. Redactable signatures allow to visibly redact parts of a signed document, while a verifier can still determine if the remaining parts were modified. If those signatures were combined with proxy re-encryption, the broker could redact unnecessary information before re-encrypting the user's attributes for a service provider, who would still be able to verify the signature for the remaining parts. For example, the user encrypts and uploads an electronic driver's license signed by a government agency. As this license contains a set of data items, the broker redacts all parts except for the user's year of birth, which is required by the service and ensured by the government's signature.

Searchable Encryption: Searchable encryption [Bon+04] would allow users to select attributes without revealing metadata about the choices to either identity broker or attribute provider. During encryption with such a searchable scheme, the ciphertexts are associated with keywords. The holder of the corresponding private key can create encrypted queries for these keywords. When given such an encrypted query, a third party storing the encrypted data is able to search for matching ciphertexts. In the context of an identity broker system, the user could tag her attributes with keywords before uploading them to the attribute provider. When the user is asked to select attributes that should be sent to the service provider, she could generate encrypted queries for the desired keywords at via her trusted device. The broker could then use these encrypted queries to retrieve matching attributes

that are the attribute provider's search results. Consequently, neither broker nor attribute provider would learn the identifying metadata about the selected encrypted attributes.

In conclusion, this thesis contributes a considerable improvement over the state of the art regarding identity management by proposing a user-friendly key management solution which enables the practical use of proxy re-encryption. However, there is plenty of opportunity to enhance the presented broker system by conducting further research in the three above mentioned directions as well as other fields of study.

9. Conclusion

In this thesis, an identity broker system was proposed and implemented, which not only tackles the security challenges arising from a cloud deployment, but also provides a usable solution to manage the involved keys. As curious cloud providers might inspect the processed data, proxy re-encryption is used to securely store and share the user's sensitive data. Furthermore, the required re-encryption keys are generated on a user-trusted cryptographic device, which is embedded through an interoperability layer.

The users' privacy was enhanced by using proxy re-encryption to handle their sensitive data in the cloud environment. Users' deposit encrypted attributes at a cloud storage provider. Proxy re-encryption allows the broker to re-encrypt these encrypted attributes for a service provider, without revealing their contents to the broker. In this thesis, a detailed analysis of multiple proxy re-encryption schemes and their properties was performed. Based on property requirements, some of these schemes proved to be suitable for the identity broker system. Eventually, the scheme by Ateniese et al. [Ate+06] was chosen, as it not only fulfills the property requirements of beeing unidirectional, collusion-safe, non-interactive, not identity-based, and CPA-secure, but also facilitates adoption as a rudimentary implementation exists. Also, the applicability of emulating proxy re-encryption with public key encryption was evaluated. However, the raised trust requirements of this emulated re-encryption exclude the broker as deployment option for the re-encryption operation.

Furthermore, a usable key management solution was proposed, which employs a cryptographic service interoperability layer (CrySIL) to integrate user-trusted devices that perform cryptographic operations with the user's key material. To re-encrypt attributes for a service provider, the broker requires a re-encryption key, which can be generated from the user's private key material. While the use of innovative cryptographic primitives in the domain of identity management has been suggested in previous works, the comfortable provisioning of key material remained an open problem. This thesis proposes to use CrySIL to expose the re-encryption key operation performed on a trusted device that holds the user's key material. With CrySIL this operation can be deployed in a multitude of locations, such as the user's phone or a trusted server. Consequently, the broker is able to obtain re-encryption keys on demand, while the user stays in control of her private key material as well as the delegation process.

Additionally, a security analysis was performed that roughly follows the Common Criteria for Information Technology Security Evaluation. In the generic part of this analysis, valuable assets were identified and multiple threats posed by various agents were considered. To protect these assets, security objectives were defined. Then, the generic analysis was applied to the implementation of the proposed identity broker system. This part of the analysis illustrated how the individual security objectives were implemented by specific countermeasures that protect the system's assets.

In summary, this thesis improved the state-of-the-art of identity management by proposing, designing and implementing a secure identity broker system that accesses the required key material for proxy re-encryption through a user-friendly key management solution which embeds user-trusted cryptographic services via an interoperability layer.

Appendix

Appendix A.

Screenshots

This section presents full-size screenshots of the implemented identity broker system used in the copyshop example. The screenshots illustrate the full process of authenticating and sharing attributes. A more detailed description of this process can be found in chapter 6, and especially in the section 6.2.

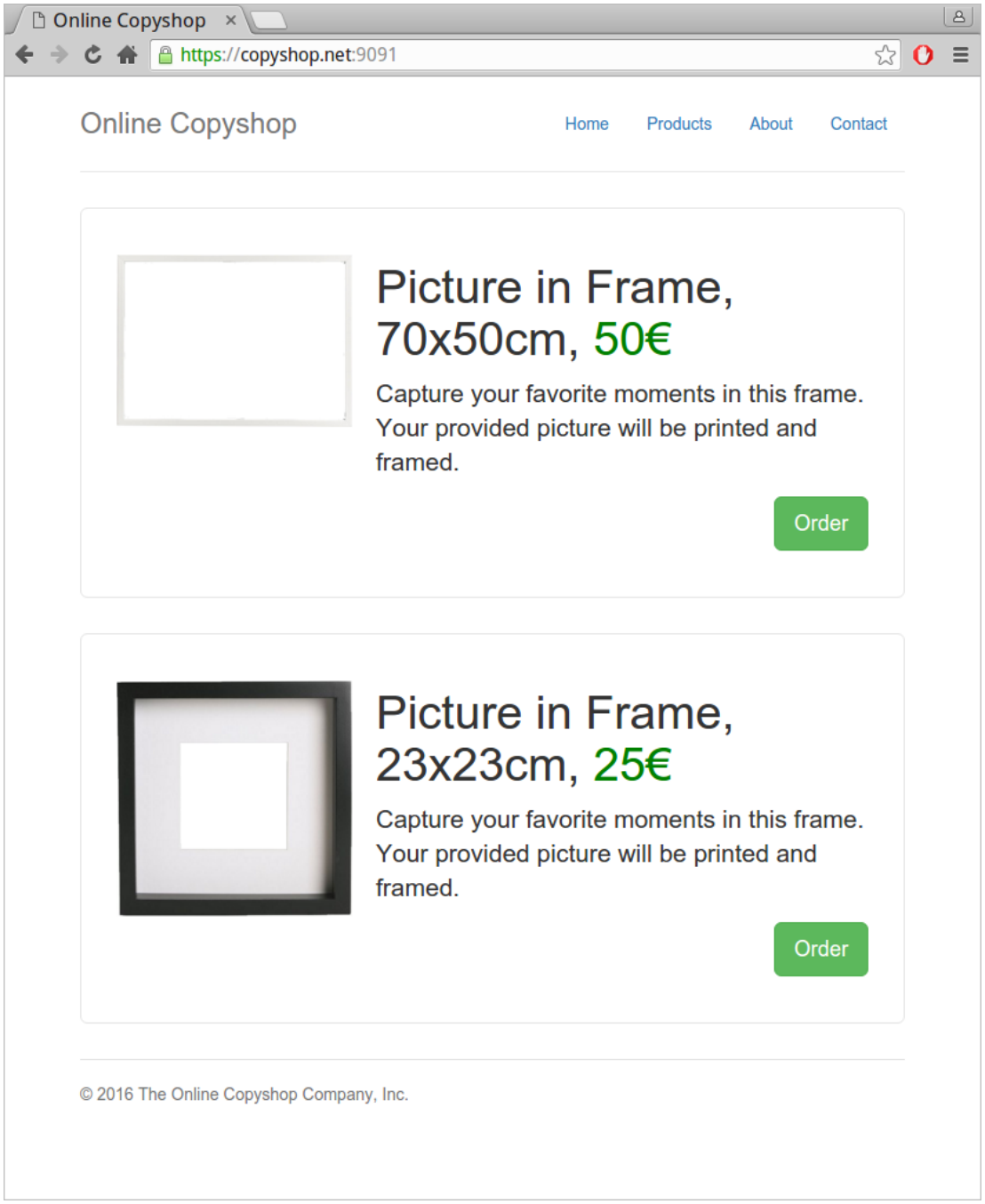


Figure A.1.: This screenshot shows the copyshop's products page. In the copyshop example, the user wants to get one of her pictures printed and framed. This page offers different options for frames to the user. A click on `Order` starts the ordering process, which requires users to authenticate and to share data.

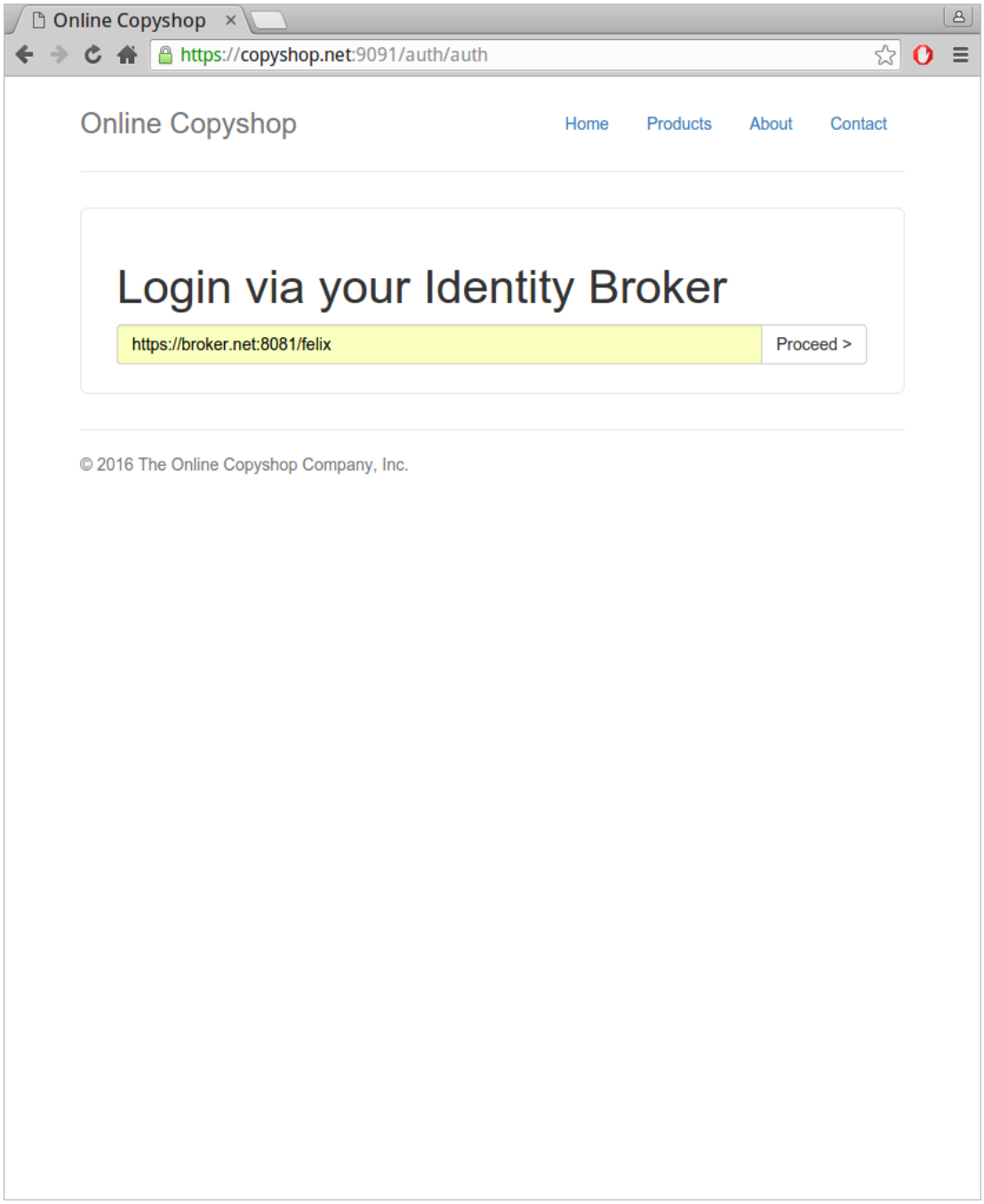


Figure A.2.: This screenshot shows the copyshop service provider asking for the user's username. This username includes a hostname, which is the basis for the identity broker discovery process. After entering the username and pressing `Proceed`, the user is forwarded to her preferred identity broker.

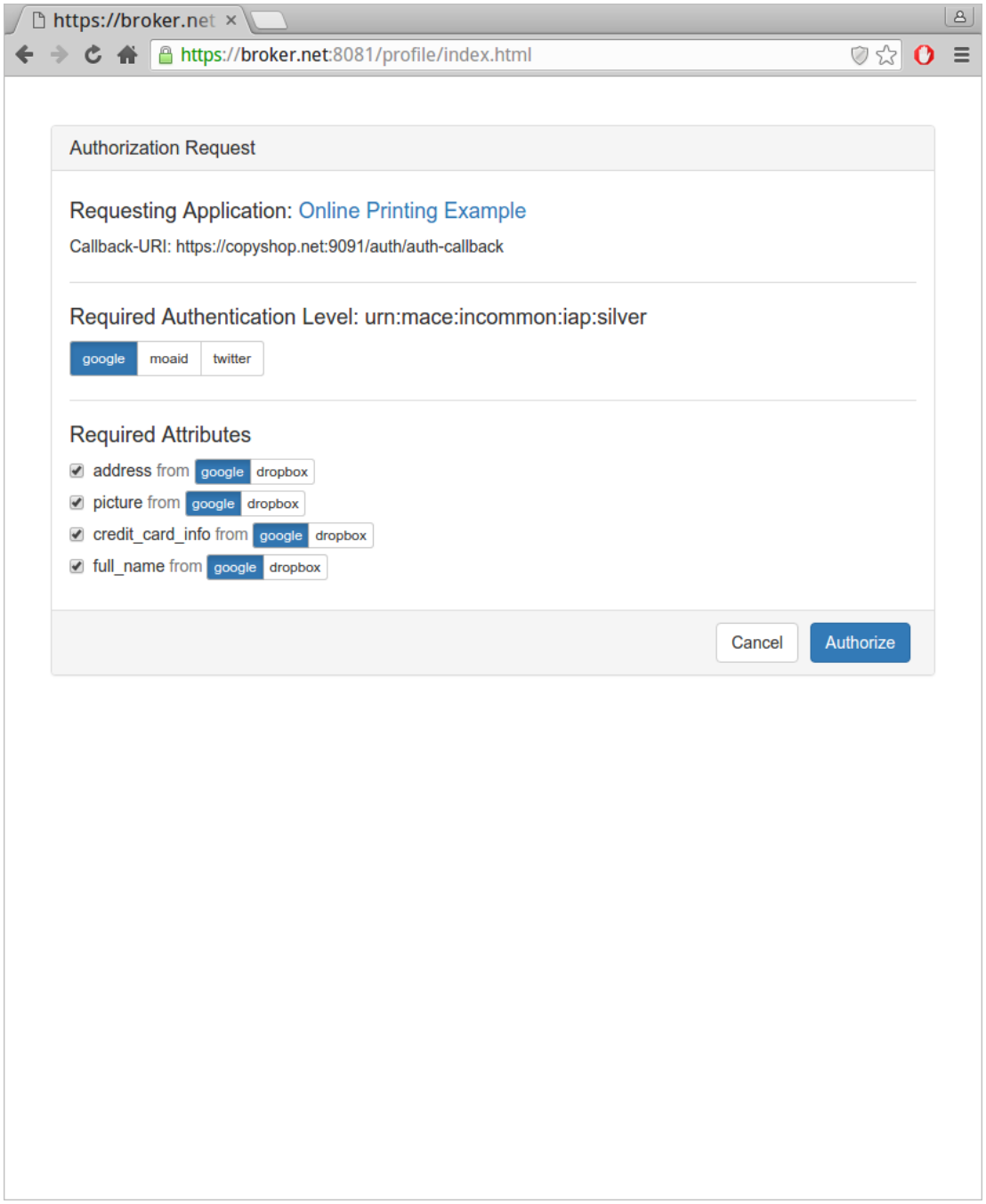


Figure A.3.: The broker presents information about the service provider's request to the user. This request includes the desired authentication level, as well as the required attributes. Based on this information, the user can either cancel the request or provide consent by selecting an identity provider and suitable attribute providers.

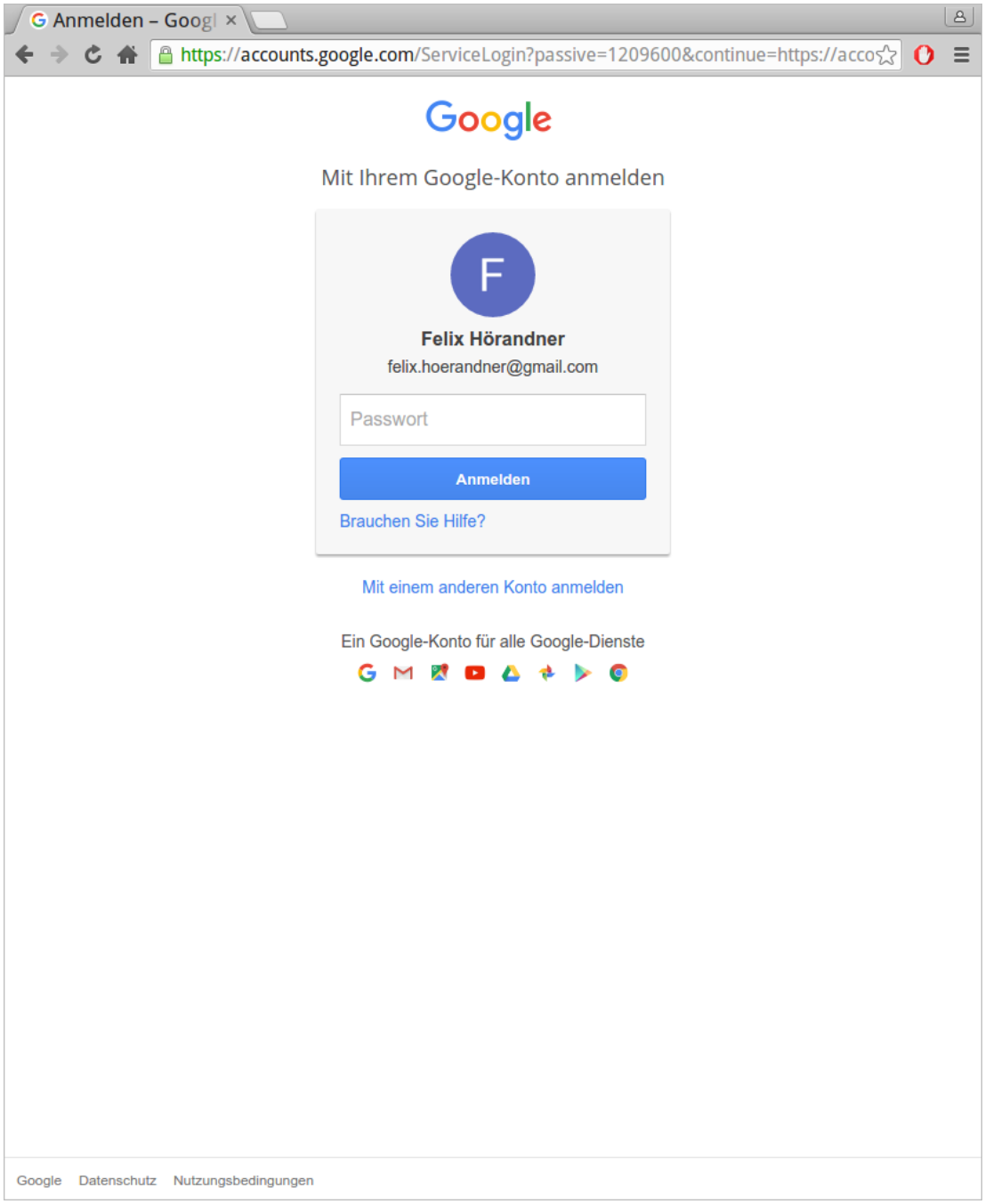


Figure A.4.: This screenshot shows the authentication form at Google as the identity provider. In this case, authentication is performed by providing a username and password pair.

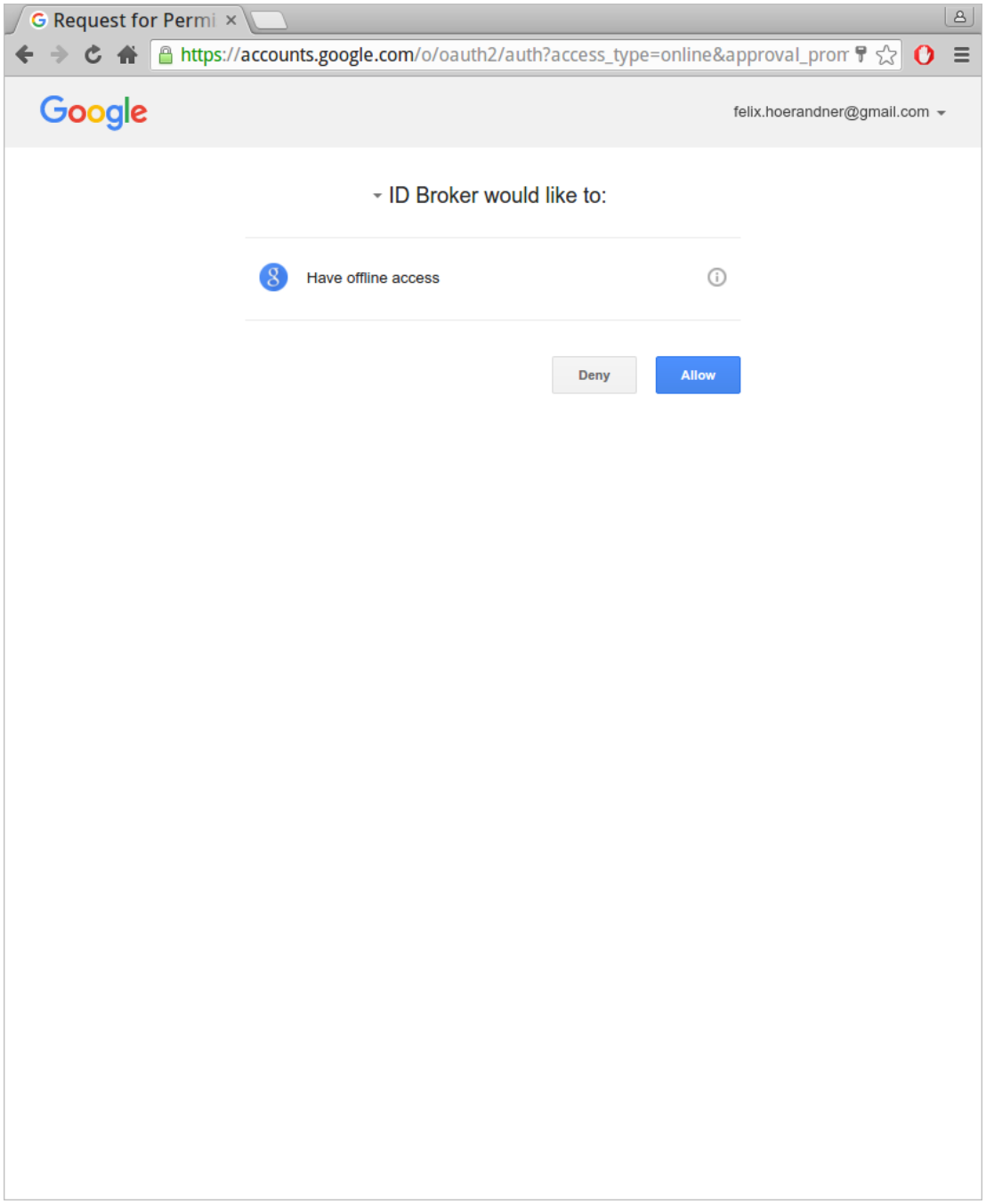


Figure A.5.: Google presents a list of requested permissions to the user. A click on `Allow` authorized the broker to get access to the user's data.

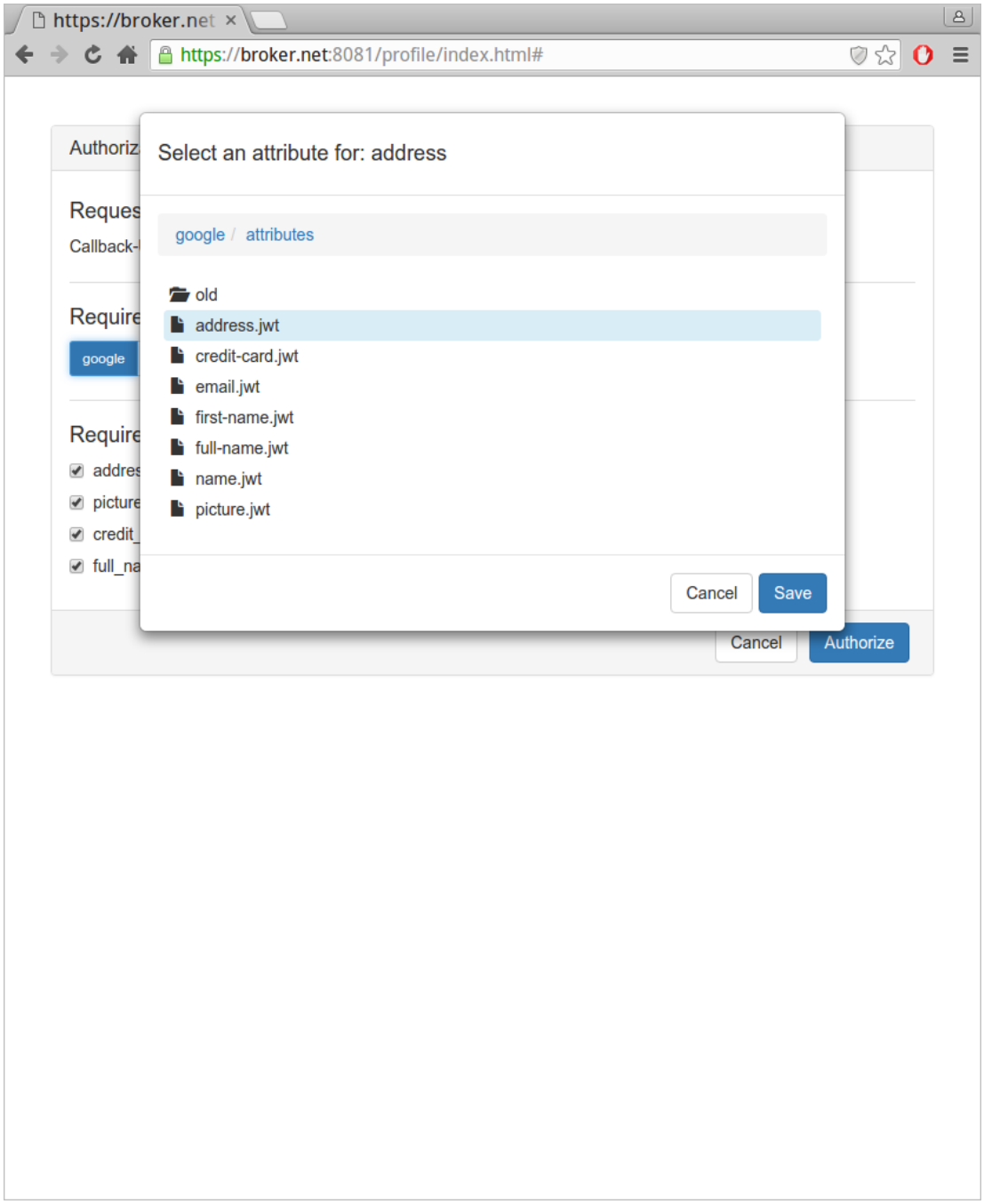


Figure A.6.: At the broker, the user is asked to select which data items at an attribute provider should be returned for the individual requested attributes. This selection process is implemented by browsing the available files at Google Drive.

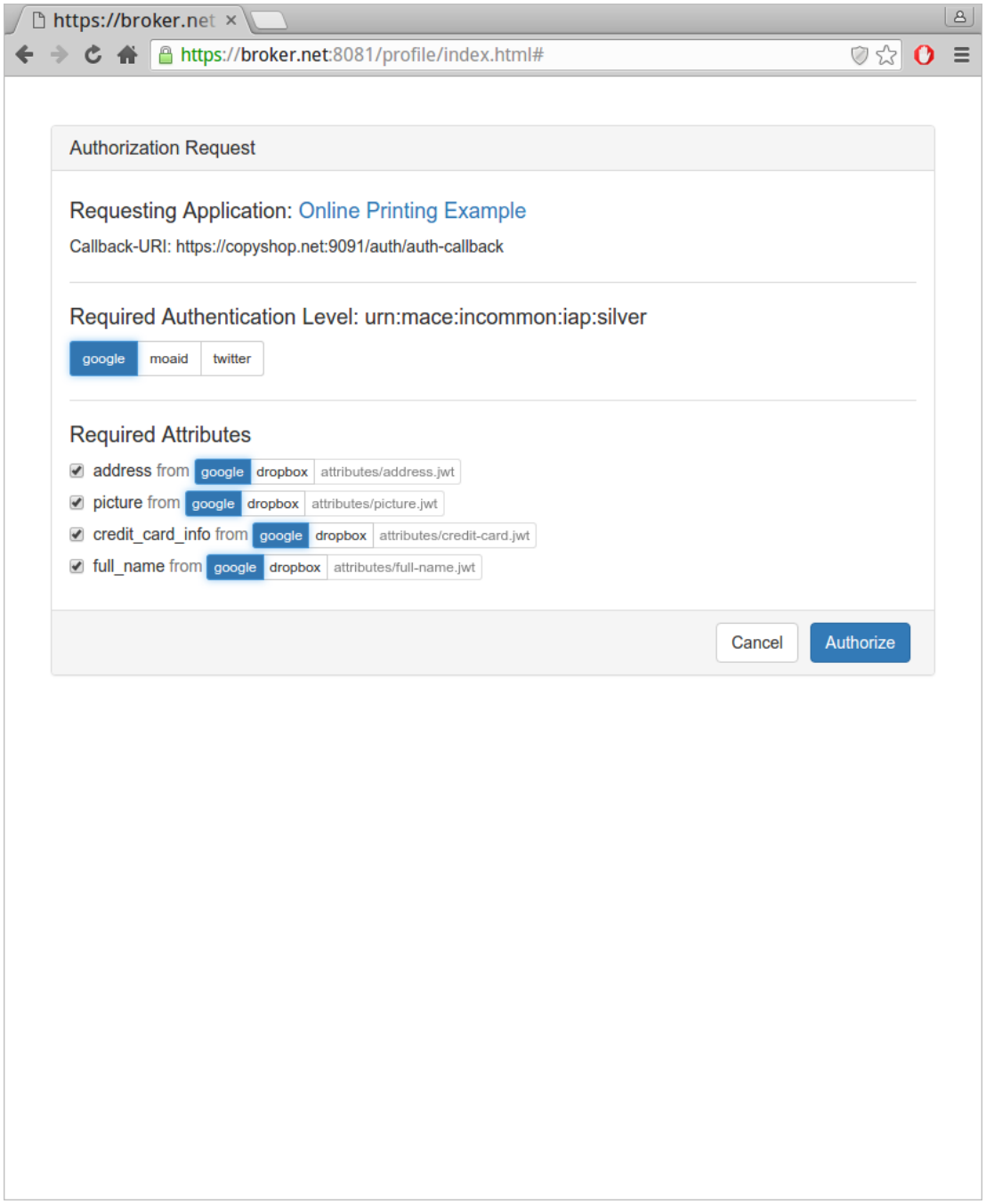


Figure A.7.: The broker presents the compiled information regarding the service provider's request to the user. After reviewing her decisions, the user can grant final consent by clicking `Authorize`.

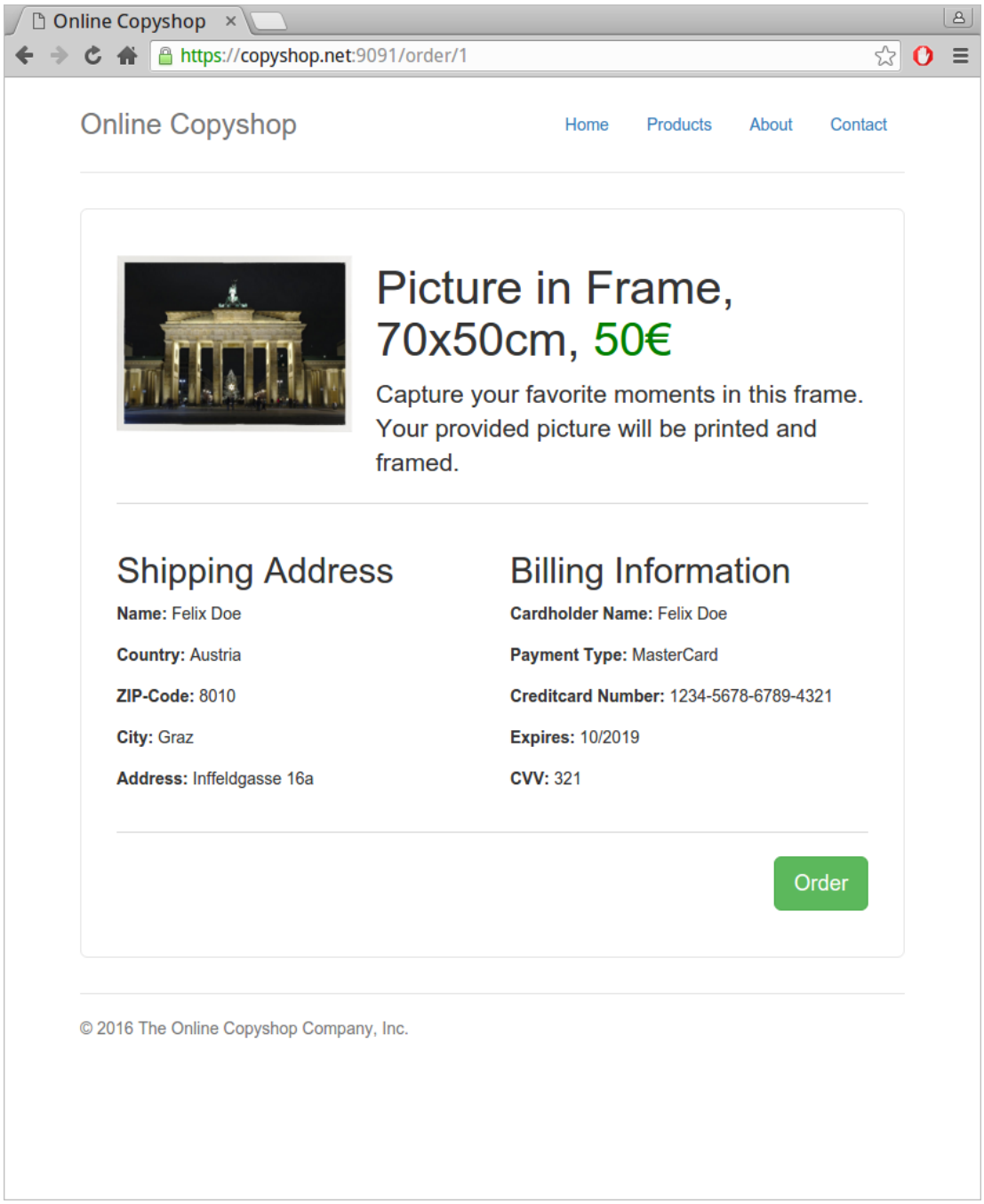


Figure A.8.: Once the sevice provider received the required information by a user completing the authentication and attribute acquisition process, the initial request can be fulfilled. This screenshot shows the ordering page of the copyshop service. This page displays the user's shared data. Name, shipping address and billing information is represented in textual form. The preview of the picture selected by the user is shown inside the chosen frame.

Bibliography

[ABH08] Giuseppe Ateniese, Karyn Benson, and Susan Hohenberger. "Key-Private Proxy Re-Encryption." In: *IACR Cryptology ePrint Archive* 2008 (2008), p. 463. URL: http://dblp.uni-trier.de/db/journals/iacr/iacr2008.html#AtenieseBH08 (cit. on pp. 32, 34, 36, 42, 43).

[Aon+13] Yoshinori Aono et al. "Key-Private Proxy Re-encryption under LWE." In: *INDOCRYPT*. Ed. by Goutam Paul and Serge Vaudenay. Vol. 8250. Lecture Notes in Computer Science. Springer, 2013, pp. 1–18. ISBN: 978-3-319-03514-7. URL: http://dblp.uni-trier.de/db/conf/indocrypt/indocrypt2013.html#AonoBPW13 (cit. on pp. 35, 36, 42).

[Ate+06] Giuseppe Ateniese et al. "Improved Proxy Re-encryption Schemes with Applications to Secure Distributed Storage." In: *ACM Trans. Inf. Syst. Secur.* 9.1 (Feb. 2006), pp. 1–30. ISSN: 1094-9224. DOI: 10.1145/1127345.1127346. URL: http://doi.acm.org/10.1145/1127345.1127346 (cit. on pp. 31, 32, 34, 36, 42, 43, 67, 93).

[BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. "Divertible protocols and atomic proxy cryptography." English. In: *Advances in Cryptology — EUROCRYPT'98*. Ed. by Kaisa Nyberg. Vol. 1403. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1998, pp. 127–144. ISBN: 978-3-540-64518-4. DOI: 10.1007/BFb0054122. URL: http://dx.doi.org/10.1007/BFb0054122 (cit. on pp. 12, 33–36, 42).

[Bon+04] Dan Boneh et al. "Public Key Encryption with Keyword Search." In: *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*. 2004, pp. 506–522. DOI: 10.1007/978-3-540-24676-3_30. URL: http://dx.doi.org/10.1007/978-3-540-24676-3_30 (cit. on p. 91).

[CH07] Ran Canetti and Susan Hohenberger. "Chosen-ciphertext Secure Proxy Re-encryption." In: *Proceedings of the 14th ACM Conference on Computer and Communications Security*. CCS '07. Alexandria, Virginia, USA: ACM, 2007, pp. 185–194. ISBN: 978-1-59593-703-2. DOI: 10.1145/1315245.1315269. URL: http://doi.acm.org/10.1145/1315245.1315269 (cit. on pp. 31, 33, 34, 36, 42).

[CT07] Cheng-Kang Chu and Wen-Guey Tzeng. "Identity-Based Proxy Re-encryption Without Random Oracles." In: *ISC*. Ed. by Juan A. Garay et al. Vol. 4779. Lecture Notes in Computer Science. Springer, Sept. 20, 2007, pp. 189–202. ISBN: 978-3-540-75495-4. URL: http://dblp.uni-trier.de/db/conf/isw/isc2007.html#ChuT07 (cit. on pp. 35, 36, 42).

[Fac16] Facebook. *Facebook Login*. 2016. URL: https://developers.facebook.com/docs/facebook-login (visited on 04/02/2016) (cit. on p. 26).

[Fan+11] Liming Fang et al. "Interactive conditional proxy re-encryption with fine grain policy." In: *Journal of Systems and Software* 84.12 (2011), pp. 2293–2302. DOI: 10.1016/j.jss.2011.06.045. URL: http://dx.doi.org/10.1016/j.jss.2011.06.045 (cit. on p. 91).

[GA07] Matthew Green and Giuseppe Ateniese. "Identity-Based Proxy Re-encryption." In: *ACNS*. Ed. by Jonathan Katz and Moti Yung. Vol. 4521. Lecture Notes in Computer Science. Springer, 2007, pp. 288–306. ISBN: 978-3-540-72737-8. URL: http://dblp.uni-trier.de/db/conf/acns/acns2007.html#GreenA07 (cit. on pp. 32, 34–36, 42).

[Goo15a] Google. *Google Drive*. 2015. URL: https://www.google.com/drive/ (visited on 02/05/2016) (cit. on p. 59).

[Goo15b] Google. *Google Identity Platform*. 2015. URL: https://developers.google.com/identity/ (visited on 02/05/2016) (cit. on pp. 26, 58).

[Har12] D. Hardt. *The OAuth 2.0 Authorization Framework*. Internet Engineering Task Force. http://tools.ietf.org/html/rfc6749. Oct. 2012 (cit. on pp. 6, 59).

[Hic15] Ian Hickson. *HTML5 Web Messaging*. W3C Recommendation. https://www.w3.org/TR/webmessaging. May 2015 (cit. on p. 12).

[Hoh+07] Susan Hohenberger et al. "Securely Obfuscating Re-encryption." English. In: *Theory of Cryptography*. Ed. by SalilP. Vadhan. Vol. 4392. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2007, pp. 233–252. ISBN: 978-3-540-70935-0. DOI: 10.1007/978-3-540-70936-7_13. URL: http://dx.doi.org/10.1007/978-3-540-70936-7_13 (cit. on pp. 34–36, 42, 43).

[Hou09] R. Housley. *Cryptographic Message Syntax (CMS)*. Internet Engineering Task Force. http://tools.ietf.org/html/rfc5652. Sept. 2009 (cit. on p. 49).

[HPS98] Jeffrey Hoffstein, Jill Pipher, and JosephH. Silverman. "NTRU: A ring-based public key cryptosystem." English. In: *Algorithmic Number Theory*. Ed. by JoeP. Buhler. Vol. 1423. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1998, pp. 267–288. ISBN: 978-3-540-64657-0. DOI: 10.1007/BFb0054868. URL: http://dx.doi.org/10.1007/BFb0054868 (cit. on p. 35).

[ID03] Anca-Andreea Ivan and Yevgeniy Dodis. "Proxy Cryptography Revisited." In: *Proceedings of the Network and Distributed System Security Symposium, NDSS 2003, San Diego, California, USA*. 2003. URL: http://www.isoc.org/isoc/conferences/ndss/03/proceedings/papers/14.pdf (cit. on p. 31).

[IEE13] IEEE. "IEEE Standard for Identity-Based Cryptographic Techniques using Pairings." In: *IEEE Std 1363.3-2013* (Nov. 2013), pp. 1–151. DOI: 10.1109/IEEESTD.2013.6662370 (cit. on p. 48).

[ISO09] ISO/IEC. *Information technology — Security techniques — Evaluation criteria for IT security*. International Standard: ISO/IEC 15408:2009, Dec. 2009 (cit. on p. 73).

[JB16] M. Jones and J. Bradley. *OpenID Connect Back-Channel Logout 1.0 - draft 02*. OpenID Foundation. http://openid.net/specs/openid-connect-backchannel-1_0.html. Feb. 2016 (cit. on p. 12).

[JBS15a] M. Jones, J. Bradley, and N. Sakimura. *JSON Web Signature (JWS)*. Internet Engineering Task Force. https://tools.ietf.org/html/rfc7515. May 2015 (cit. on p. 5).

[JBS15b] M. Jones, J. Bradley, and N. Sakimura. *JSON Web Token (JWT)*. Internet Engineering Task Force. https://tools.ietf.org/html/rfc7519. May 2015 (cit. on pp. 6, 49).

[JCM15] M. Jones, B. Campbell, and C. Mortimore. *JSON Web Token (JWT) Profile for OAuth 2.0 Client Authentication and Authorization Grants*. Internet Engineering Task Force. http://tools.ietf.org/html/rfc7523. May 2015 (cit. on p. 9).

[JH12] M. Jones and D. Hardt. *The OAuth 2.0 Authorization Framework: Bearer Token Usage*. Internet Engineering Task Force. http://tools.ietf.org/html/rfc6750. Oct. 2012 (cit. on p. 7).

[JH15] M. Jones and J. Hildebrand. *JSON Web Encryption (JWE)*. Internet Engineering Task Force. https://tools.ietf.org/html/rfc7516. May 2015 (cit. on p. 6).

[Joh+02] Robert Johnson et al. "Topics in Cryptology — CT-RSA 2002: The Cryptographers' Track at the RSA Conference 2002 San Jose, CA, USA, February 18–22, 2002 Proceedings." In: ed. by Bart Preneel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002. Chap. Homomorphic Signature Schemes, pp. 244–262. ISBN: 978-3-540-45760-2. DOI: 10.1007/3-540-45760-7_17. URL: http://dx.doi.org/10.1007/3-540-45760-7_17 (cit. on p. 91).

[Jon+13] P. Jones et al. *WebFinger*. Internet Engineering Task Force. http://tools.ietf.org/html/rfc7033. Sept. 2013 (cit. on p. 10).

[Jon15a] M. Jones. *JSON Web Algorithms (JWA)*. Internet Engineering Task Force. https://tools.ietf.org/html/rfc7518. May 2015 (cit. on p. 5).

[Jon15b] M. Jones. *JSON Web Key (JWK)*. Internet Engineering Task Force. https://tools.ietf.org/html/rfc7517. May 2015 (cit. on p. 5).

[Jon16] M. Jones. *OpenID Connect Front-Channel Logout 1.0 - draft 00*. OpenID Foundation. http://openid.net/specs/openid-connect-frontchannel-1_0.html. Feb. 2016 (cit. on p. 12).

[LV08] Benoît Libert and Damien Vergnaud. "Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption." English. In: *Public Key Cryptography – PKC 2008*. Ed. by Ronald Cramer. Vol. 4939. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 360–379. ISBN: 978-3-540-78439-5. DOI: 10.1007/978-3-540-78440-1_21. URL: http://dx.doi.org/10.1007/978-3-540-78440-1_21 (cit. on pp. 33, 34, 36, 42, 43).

[Med+16] B. de Medeiros et al. *OpenID Connect Session Management 1.0 - draft 26*. OpenID Foundation. http://openid.net/specs/openid-connect-session-1_0.html. Feb. 2016 (cit. on p. 12).

[NAL12] David Nuñez, Isaac Agudo, and Javier Lopez. "Integrating OpenID with proxy re-encryption to enhance privacy in cloud-based identity services." In: *4th IEEE International Conference on Cloud Computing Technology and Science Proceedings, CloudCom 2012, Taipei, Taiwan, December 3-6, 2012*. 2012, pp. 241–248. DOI: 10.1109/CloudCom.2012.6427551. URL: http://dx.doi.org/10.1109/CloudCom.2012.6427551 (cit. on p. 1).

[NAL15] David Nuñez, Isaac Agudo, and Javier Lopez. "NTRUReEncrypt: An Efficient Proxy Re-Encryption Scheme Based on NTRU." In: *ASIACCS*. Ed. by Feng Bao et al. ACM, 2015, pp. 179–189. ISBN: 978-1-4503-3245-3. URL: http://dblp.uni-trier.de/db/conf/ccs/asiaccs2015.html#NunezAL15 (cit. on pp. 35, 36, 42).

[Rei+15] Florian Reimair et al. "MoCrySIL - Carry Your Cryptographic Keys in Your Pocket." In: *SECRYPT 2015 - Proceedings of the 12th International Conference on Security and Cryptography, Colmar, Alsace, France, 20-22 July, 2015*. 2015, pp. 285–292. DOI: 10.5220/0005547902850292. URL: http://dx.doi.org/10.5220/0005547902850292 (cit. on pp. 14, 15).

[RTZ15] Florian Reimair, Peter Teufl, and Thomas Zefferer. "WebCrySIL - Web Cryptographic Service Interoperability Layer." In: *WEBIST 2015 - Proceedings of the 11th International Conference on Web Information Systems and Technologies, Lisbon, Portugal, 20-22 May, 2015*. 2015, pp. 35–44. DOI: 10.5220/0005488400350044. URL: http://dx.doi.org/10.5220/0005488400350044 (cit. on p. 14).

[Sak+14] Nat Sakimura et al. *OpenID Connect Core 1.0 incorporating errata set 1*. OpenID Foundation. http://openid.net/specs/openid-connect-core-1_0.html. Nov. 2014 (cit. on p. 6).

[Sho97] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer." In: *SIAM J. Comput.* 26.5 (Oct. 1997), pp. 1484–1509. ISSN: 0097-5397. DOI: 10.1137/S0097539795293172. URL: http://dx.doi.org/10.1137/S0097539795293172 (cit. on p. 32).

[Wen+09] Jian Weng et al. "Efficient Conditional Proxy Re-encryption with Chosen-Ciphertext Security." In: *Information Security, 12th International Conference, ISC 2009, Pisa, Italy, September 7-9, 2009. Proceedings*. 2009, pp. 151–166. DOI: 10.1007/978-3-642-04474-8_13. URL: http://dx.doi.org/10.1007/978-3-642-04474-8_13 (cit. on p. 91).

[XT10] Keita Xagawa and Keisuke Tanaka. "Proxy re-encryption based on learning with errors." In: *Proceedings of the 2010 Symposium on Cryptography and Information Security (SCIS 2010)*. 2010 (cit. on pp. 32, 35, 36, 42).

[ZS13] Bernd Zwattendorfer and Daniel Slamanig. "Privacy-Preserving Realization of the STORK Framework in the Public Cloud." In: *10th International Conference on Security and Cryptography (SECRYPT 2013), Reykjavik, Iceland, 29-31 July 2013*. 2013, pp. 419–426 (cit. on p. 1).

[Zwa+14] Bernd Zwattendorfer et al. "A Federated Cloud Identity Broker-Model for Enhanced Privacy via Proxy Re-Encryption." In: *15th IFIP TC6/TC11 International Conference on Communications and Multimedia Security, CMS'2014, September 25th - 26th, 2014, Aveiro, Portugal*. 2014, pp. 92–103 (cit. on p. 1).