Towards Privacy-Preserving and User-Centric Identity Management as a Service

Pritam Dash[1], Christof Rabensteiner [2], Felix Hörandner [3], Simon Roth [4]

Abstract: Identification, authentication and the exchange of users' identity information are key factors in protecting access to online services. Especially cost-effectiveness is a considerable incentive to move identity management models into the public cloud. As cloud environments are not fully trusted, the users' sensitive attributes must not be stored or transmitted in plain, while it still has to be possible to share them. One approach is to employ proxy re-encryption, which enables the identity provider to transform a user's encrypted attributes into ciphertext for an authorized service provider. However, for adoption, the user's perspective must not be neglected. In this paper, we propose a user-friendly and user-centric identity management solution that employs cryptographic mechanisms to protect the users' privacy and keep them in control of the data sharing process. We integrate proxy re-encryption into the widely-adopted OpenID Connect protocol to achieve end-to-end confidentiality. To make this concept user-friendly, we introduce a mobile app that handles the involved cryptographic operations which rely on keys securely stored in a trusted execution environment.

Keywords: Identity management, OpenID Connect, Cloud Computing, Mobile Application, Proxy Re-Encryption, Trusted Execution Environment.

1 Introduction

In identity management (IdM) solutions, service providers (SPs) outsource identification and authentication processes to an identity provider (IdP). Initially, identity management was mainly limited to inter-organizational approaches within a single domain. Over time, more sophisticated federated approaches emerged, which enabled identification and authentication across organizational domains or even national borders [LZ10] through standardized protocols such as OpenID Connect [Sa14], SAML [Ca09], or WS-Federation [GN09]. The benefits of cloud computing, such as reduced costs and scalability, also represent convincing incentives for identity management systems. By moving the identity management processes into the cloud, Identity (and Access) Management can be offered "as a Service" (IDaaS) to applications operated both in closed domains and in the cloud.

[1] Stiftung Secure Information and Communication Technologies (SIC), Inffeldgasse 16a, 8010 Graz, Austria, pritam.dash@iaik.tugraz.at

[2] Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology (TUG), Inffeldgasse 16a - Second F, 8010 Graz, Austria, christof.rabensteiner@iaik.tugraz.at

[3] Institute for Applied Information Processing and Communications (IAIK), Graz University of Technology (TUG), Inffeldgasse 16a - Second F, 8010 Graz, Austria, felix.hoerandner@iaik.tugraz.at

[4] Stiftung Secure Information and Communication Technologies (SIC), Inffeldgasse 16a, 8010 Graz, Austria, simon.roth@iaik.tugraz.at

While traditional identity providers are usually deployed in-house and considered to be fully trusted with respect to privacy, this assumption does not hold anymore in the cloud [Hö16]. Since cloud identity providers are able to inspect the deployed applications and their data, they are considered to be only semi-trusted and acting honest-but-curious [CS10]. Therefore, privacy and data confidentiality represent significant challenges, especially when dealing with sensitive information. To overcome the privacy issues of existing IdM systems in public clouds, one solution is to use proxy re-encryption with established identity protocols. Proxy re-encryption extends asymmetric encryption by enabling a semi-trusted proxy to transform encrypted data of one entity into ciphertext for another without learning the underlying message. Thereby secure and privacy-preserving sharing of information between users and service providers in cloud-based applications is possible.

In related work, several identity management concepts based on proxy re-encryption have been proposed. Nunez et al. enhanced OpenID [NAL12] and SAML [NA14] with proxy re-encryption such that data-curious identity providers do not learn the user's identity attributes. However, [NA14] moves control over the data sharing away from the user to a representative entity. One of the major setbacks here is that the private key of the representative entity is used to generate re-encryption keys for all the users it represents. Zwattendorfer et al. [Zw14] proposed a federated cloud identity broker model employing proxy re-encryption to achieve enhanced privacy. In [ZS15], they propose three approaches on how to move aspects of the eID infrastructure into the public cloud and in [ZS16] further extend the proposed design in the context of the Austrian eID solution. Their approach leverages proxy re-encryption and redactable signatures [Jo02] to protect users against privacy invasions. While the above mentioned papers and their core idea of using proxy re-encryption in identity management build the basis of our approach, they do not address the usability, security and trust challenges associated with certain processes such as: allowing the user to conveniently encrypt data, generating re-encryption keys for authorized service providers, and exporting such data to the cloud.

In this paper, we propose a more complete identity management solution that aims to be privacy-preserving, user-centric and user-friendly. Our contribution consists of two parts:

We provide guidelines on how to integrate proxy re-encryption into the widely adopted OpenID Connect ecosystem to ensure end-to-end data confidentiality. Users stay in control of the data sharing process as they have to generate a re-encryption key to enable that their encrypted identity data is re-encrypted for an authorized service.

We introduce a User Identity Management (UIM) App for smartphones to implement our vision in a user-friendly way. This app represents a trusted domain to conveniently perform the cryptographic operations required on the user's side. Independent from the device used to initiate the identification and authentication process, the user interacts with the UIM app on her nowadays typically omnipresent smartphone to cryptographically authorize the data sharing process. Also, this app can leverage features of modern phones, such as the trusted execution environment that enables to implement proxy re-encryption while securely storing the keys bound to the hardware.

This paper is organized as follows: In Section 2, we briefly discuss the background technologies used in our proposed IDaaS system: OpenID Connect, proxy re-encryption and the trusted execution environment. In Section 3, we present our proposed IDaaS system, its architecture, data flow and trust model. Section 4 elaborates on the integration of proxy re-encryption into the OpenID Connect ecosystem. Section 5 presents how our concept can be implemented in a user-friendly way with our User Identity Management App. In Section 6, we discuss the model based on our objectives. Finally, in Section 7 we conclude this paper and present topics for future research.

2 Background Technologies

OpenID Connect: OpenID Connect [Sa14] is an identity layer on top of OAuth 2.0 and specifies authentication and authorization of users in federated identity management models. Using OpenID Connect, the SP can delegate the authentication to an IdP. OpenID Connect offers different authentication flows; the most common one is the *authorization code flow* as seen in Figure 1. By the end of this flow, the user is authenticated towards the SP. Knowing the identity of the user allows the SP to accept or deny access to the protected resource.

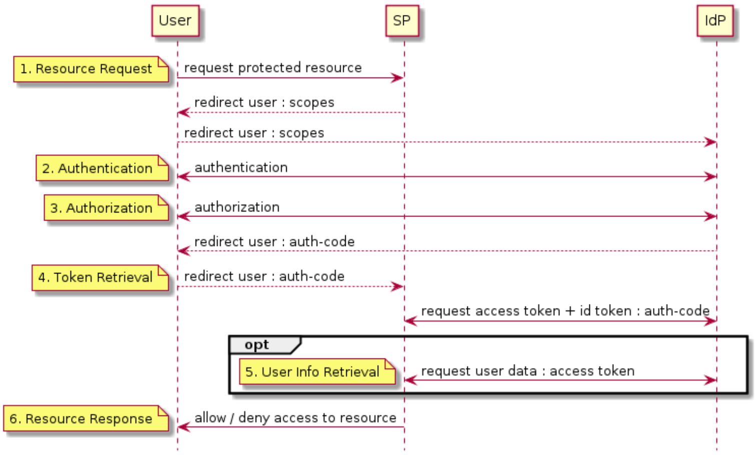

Fig. 1: OpenID Connect Authorization Code Flow

The authorization code flow in Figure 1 goes through the following phases: In Phase 1, the unauthenticated user requests a protected resource, but the SP is not yet able to decide if she is allowed to access. Therefore, the SP delegates the authentication process to the IdP (Phase 2) by redirecting the user to the IdP. When redirecting the user, the SP specifies which identity attributes are needed to reach an access decision. The SP groups these attributes in so-called *scopes* and attaches them to the redirect. During authorization (Phase 3), the

IdP asks for the user's consent to share the requested data with the SP. After successful authentication and authorization, the IdP redirects the user back to the SP. With this redirect, the IdP passes along the *auth-code*. The SP uses this auth-code to retrieve the access token and the ID token directly from the IdP (Phase 4). The access token allows the SP to fetch user data from the IdP, whereas the ID token identifies the user. In an optional Phase 5, the SP may retrieve additional user information with the access token. Given the ID token and user information, the SP can now respond to the initial resource request in Phase 6.

Proxy Re-Encryption: Proxy re-encryption [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. The applications of proxy re-encryption are manifold. Examples are forwarding encrypted emails or sharing encrypted files.

$$\texttt{KeyGen}(\kappa) \to (sk_A, pk_A) \tag{1}$$

$$\texttt{Enc}(M, pk_A) \to C_A \tag{2}$$

$$\texttt{Dec}(C_A, sk_A) \to M \tag{3}$$

Basic asymmetric encryption can be described with three algorithms. `KeyGen` generates a key pair containing a private key sk_A and a public key pk_A (Eq. 1). Given a public key pk_A and a message M, `Enc` encrypts the message into a ciphertext C_A for the owner of the key pair to which pk_A belongs (Eq. 2). On input of a private key sk_A and a ciphertext C_A, `Dec` decrypts the ciphertext C_A into M if C_A was encrypted for the user who owns the key pair that includes sk_A (Eq. 3).

$$\texttt{ReKeyGen}(sk_A, pk_B) \to rk_{A\to B} \tag{4}$$

$$\texttt{ReEnc}(C_A, rk_{A\to B}) \to C_B \tag{5}$$

Proxy re-encryption introduces two additional operations. `ReKeyGen` takes the private key sk_A of Alice and some key of Bob to create a re-encryption key $rk_{A\to B}$ (Eq. 4). Given a re-encryption key $rk_{A\to B}$, `ReEnc` transforms the ciphertext C_A decryptable by Alice into a ciphertext C_B decryptable by Bob (Eq. 5). Depending on the concrete scheme, re-encrypted ciphertexts can be re-encrypted again (multi-hop) or not (single-hop).

Trusted Execution Environment: The Trusted Execution Environment (TEE) is an execution environment that offers confidential and integrity protected data storage and executes *trusted applications*. These trusted applications are isolated from the regular execution environment of the rich OS and can only be accessed through well-defined channels. Many of today's mobile devices support the execution of trusted applications on TEE.

3 System Architecture

A typical identity management scenario involves a user that wants to access a protected resource at a service provider. To facilitate access and to lower the efforts, the service provider outsources the authentication process to an identity provider. In this section, we describe the architecture of our IDaaS solution. First, we introduce the main actors and discuss their setup and interactions in an identity management process utilizing proxy re-encryption. Then, we discuss the trust relationships and assumptions of our system.

Actors: Our system consists of three actors: the *service provider* (*SP*), the *identity provider* (*IdP*), and our *User Identity Management App* (*UIM app*). The service provider allows users to access protected resources or functions from their web browser or mobile app. The identity provider is responsible for authentication and authorization. The UIM app enables the interaction between the user and the IdP and handles the user's involved cryptographic keys. Figure 2 gives an overview of these actors and their interactions.

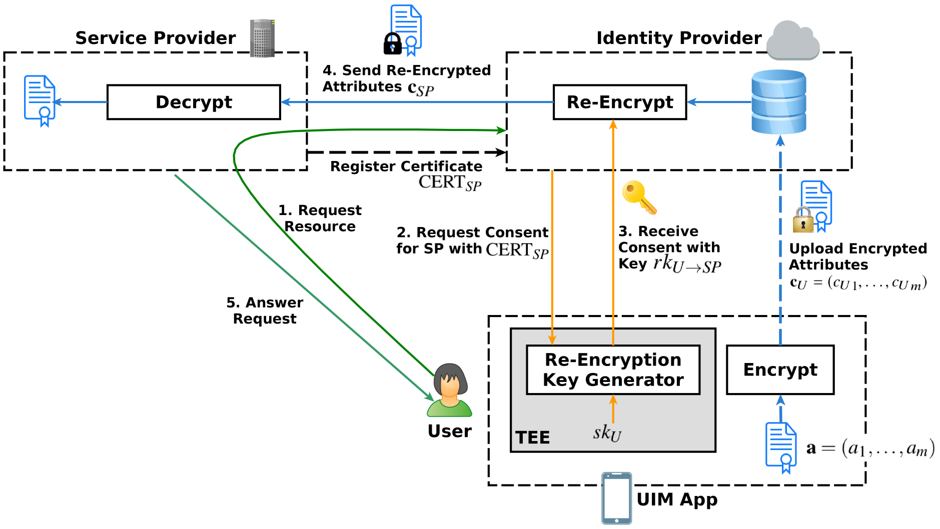
Service Provider
Decrypt
4. Send Re-Encrypted Attributes $\mathbf{c}_{SP}$
Identity Provider
Re-Encrypt
Register Certificate CERT_{SP}
1. Request Resource
2. Request Consent for SP with CERT_{SP}
3. Receive Consent with Key $rk_{U \rightarrow SP}$
Upload Encrypted Attributes $\mathbf{c}_U = (c_{U\,1}, \ldots, c_{U\,m})$
5. Answer Request
User
Re-Encryption Key Generator
TEE
sk_U
Encrypt
$\mathbf{a} = (a_1, \ldots, a_m)$
UIM App

Fig. 2: High Level Architecture of our Identity Management Concept

Process: For the IdP to be able to perform such delegated authentication and authorization and encrypt user data end-to-end, the user has to register at the IdP in a preliminary step. During registration, the UIM app generates the user's key pair, encrypts the user's identity attributes with her public key, and stores the encrypted attributes at the IdP. The identity management process visualized in Figure 2 is described in the following steps:

Step 1: The user tries to access a protected resource at a service provider. This service provider has to make an authorization decision based on the user's identity. To acquire the required data, the service provider redirects the user to her preferred IdP.

Step 2: The IdP requests the user's consent to share data with the SP. Therefore, the IdP sends a push notification to the UIM app that includes the SP's certified public key pk_{SP} as well as a list of required attributes. In the UIM app, the user reviews this list and selects the particular identity attributes she wants to share with the SP.

Step 3: If the user agrees to share data, the *Re-Encryption Key Generator* component of the UIM app generates a re-encryption key $rk_{U \to SP}$ using the SP's public key pk_{SP} and the user's private key sk_U. This re-encryption key is sent to the IdP to re-encrypt the selected attributes for the SP.

Step 4: With the re-encryption key $rk_{U \to SP}$, the IdP transforms the selected identity attributes $\mathbf{c}_U = (c_{U1}, \ldots, c_{Un})$ into $c_{SP} = \texttt{ReEnc}(\mathbf{c}_U, rk_{U \to SP})$ and returns them to the SP.

Step 5: The SP receives the resulting proof of authentication and the re-encrypted attributes c_{SP}. Then, the SP decrypts these attributes with her private key sk_{SP} and is finally able to either accept or deny access to the originally requested resource.

Trust Model: This section describes the trust relationships between the user and the other actors. The user trusts the IdP to broker her attributes, but does not want the IdP to know their plain content. We assume that the IdP acts *honest but curious*, which means that it fulfills all protocol requirements, but may perform passive attacks to learn about the user's sensitive data to, for example, sell attributes to third parties. We exclude active attacks in our trust model, since they are more likely to be detected by other parties and thus less appealing. The user fully trusts the UIM app to handle her identity attributes and private keys, and to not disclose them to third parties. The app itself can be seen as trustworthy because it runs on the user's device and is therefore under her control. Besides the aspect of physical ownership, the trust in the app boils down to trusting third party code in general, which is out of scope of this paper. The user trusts the SP to learn and process the user-selected attributes, but not necessarily all of her data. Furthermore, we assume that the SP does not collude with the IdP.

4 Integration of Proxy Re-Encryption into OpenID Connect

This section explains how we adapt OpenID Connect such that we preserve the user's privacy while keeping interactions user-friendly and user-centric. First, we introduce the re-encryption key generator and state how cryptographic material is being exchanged prior to the OpenID Connect authentication sequence. Then, we describe how we adapt the authentication sequence from Figure 1. Finally, this section discusses the format of the exchanged data. Table 1 explains the symbols used in the remainder of this section.

Key Generator: The *re-encryption key generator* (KG) participates in the OpenID Connect authentication sequence and generates the re-encryption key $rk_{U \to SP}$ for the IdP upon request. To generate this re-encryption key, the KG requires access to sk_U and therefore has to be fully trusted. The KG generates the $rk_{U \to SP}$ *if and only if* the user agrees to share her attributes with the presented SP. This condition gives the user control over her attributes, because it prevents the IdP from requesting re-encryption keys for not approved parties. The

U	user	n_i	attribute name
v_i	attribute value	$a_i = (id_a, \mathsf{n}_i, v_i)$	attribute tuple
id_a	att. set identifier	$\sigma_I(a_i)$	signature of a_i by I
I	issuer	$\mathbf{a} = (a_i, \sigma_I(a_i))_{1 \leq i \leq m}$	U's bundled attributes
pk_U	U's public key	$c_{Ui} = \mathtt{Enc}((a_i, \sigma_I(a_i)), pk_U)$	a_i, signed by I, encrypted by U
sk_U	U's private key	$c_{SPi} = \mathtt{Enc}((a_i, \sigma_I(a_i)), pk_{SP})$	a_i, signed by I, encrypted by SP
pk_{SP}	SP's public key	$\mathbf{c}_U = (c_{U1}, \ldots, c_{Um})$	attributes, encrypted by U
$rk_{U \to SP}$	re-encryption key	$\mathbf{c}_{SP} = (c_{SP1}, \ldots, c_{SPn})$	attributes, encrypted by SP
CERT$_{SP}$	certificate of SP	$\mathbf{t} = (t_1, \ldots, t_r)$	transactional information

Tab. 1: Symbols and Explanation

KG can be implemented in different ways, e.g. as a server within a company network or as a browser extension. In our scenario, the KG runs in the TEE of a mobile device.

Setup: Before the authentication sequence can take place, we ensure that each party possesses the necessary cryptographic material, namely pk_{SP} and $\mathbf{c}_U$. Firstly, the SP has to provide pk_{SP} to the IdP, as the IdP needs this key when requesting $rk_{U \to SP}$ from the KG. Concretely, the exchange of pk_{SP} between SP and IdP can be achieved through the *Dynamic Client Registration* specification [SBJ11], which offers a dedicated parameter for key exchanges within the *Client Metadata*. In order to protect origin and integrity of pk_{SP}, we wrap it in the certificate CERT$_{SP}$, which is issued by a trusted third party. The certificate contains identity information about the requesting SP thath helps the user decide with whom she wants to her share data. Secondly, prior to the authentication sequence, the user encrypts both attributes and corresponding signatures into $\mathbf{c}_U$. These attributes can either be signed by the user (self-issued) or by a third party issuer. The user then deposits $\mathbf{c}_U$ at the IdP, so that the IdP can re-encrypt and share the identity attributes during the authentication sequence.

Authentication Sequence Adaption: We integrate changes into the OpenID Connect authentication sequence from Figure 1 and depict these changes in Figure 3. The changes affect the *authorization phase*, where the IdP requests $rk_{U \to SP}$, and the *token retrieval phase*, where the IdP and the SP handle and exchange the encrypted attributes.

In the *authorization phase*, the IdP asks the user if and which *scopes* can be shared with the SP. We alter this phase by tying in the request for $rk_{U \to SP}$. This way, we combine both questions of authorization ("Which scopes can be shared?" and "Can we generate a re-encryption key for SP?") into one single user interaction. While asking for consent, the IdP passes both the CERT$_{SP}$ and the scopes to the KG (Figure 3, I). The KG verifies the integrity and authenticity of pk_{SP} (II) and presents both identity of SP and requested scopes to the user when asking for consent ("Would you like to share <scopes> with SP?") in Step III. The user reviews the request and agrees to share the requested scopes with SP (IV). With the user's permission, the KG generates $rk_{U \to SP} = \mathtt{ReKeyGen}(sk_U, pk_{SP})$ in

Step V and returns $rk_{U \rightarrow SP}$ to the IdP (VI). In the *token retrieval phase*, the IdP re-encrypts and forwards the user's attributes to the SP. In Step VII, the IdP selects the authorized attributes from $\mathbf{c}_U$ and transforms them into $\mathbf{c}_{SP} = \mathtt{ReEnc}(\mathbf{c}_U, rk_{U \rightarrow SP})$. The IdP combines the re-encrypted attributes $\mathbf{c}_{SP}$ with transactional information $\mathbf{t}$ such as, the issue time and intended audience. This combined data and its signature $\sigma_{IdP}(\mathbf{t} \cup \mathbf{c}_{SP})$ is then forwarded to the SP (VIII). After receiving $\mathbf{c}_{SP}$, the SP uses its private key sk_{SP} to decrypt the signed attributes $\mathbf{a} = \mathtt{Dec}(\mathbf{c}_{SP}, sk_{SP})$ (IX). Step VIII and Step XI happen when the ID token is retrieved or data is fetched from the `UserInfo Endpoint`.

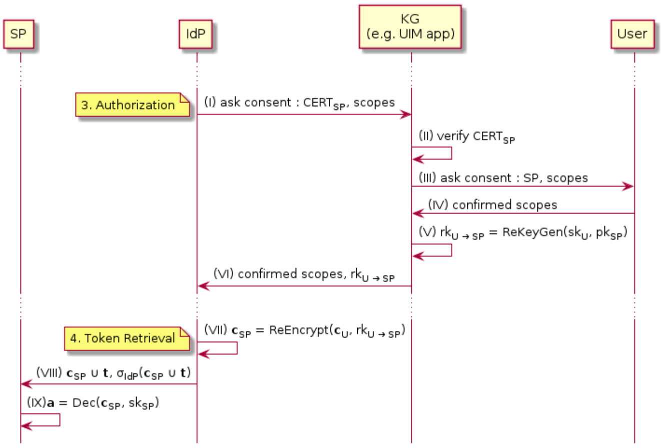
SP
IdP
KG
(e.g. UIM app)
User
3. Authorization
(I) ask consent : CERT$_{SP}$, scopes
(II) verify CERT$_{SP}$
(III) ask consent : SP, scopes
(IV) confirmed scopes
(V) $rk_{U \rightarrow SP}$ = ReKeyGen(sk_U, pk_{SP})
(VI) confirmed scopes, $rk_{U \rightarrow SP}$
4. Token Retrieval
(VII) $\mathbf{c}_{SP}$ = ReEncrypt($\mathbf{c}_U$, $rk_{U \rightarrow SP}$)
(VIII) $\mathbf{c}_{SP} \cup \mathbf{t}$, $\sigma_{IdP}(\mathbf{c}_{SP} \cup \mathbf{t})$
(IX)$\mathbf{a}$ = Dec($\mathbf{c}_{SP}$, sk_{SP})

Fig. 3: OpenID Connect Authorization Code Flow with Proxy Re-Encryption Integration

Format: The parameters exchanged in Figure 3 are structured and encoded as follows: CERT$_{SP}$ is encoded as a X.509 certificate and $rk_{U \rightarrow SP}$ as a `JSON Web Key` (`JWK`). `JSON Web Signatures` (`JWS`) are used for individual attribute tuples a_i and for the set $\mathbf{t} \cup \mathbf{c}_{SP}$. Each encrypted identity attribute c_{U_i} (resp. c_{SP_i}) is wrapped in `JSON Web Encryption` (`JWE`) by itself. Signing and encrypting attributes *individually* allows the IdP to exclusively select attributes that have been authorized by the user and to re-encrypt them for the SP. The attribute set identifier id_a is shared between bundled attributes (e.g. name and birthday) and prevents a user from combining attributes that do not belong together. We use a hybrid scheme between proxy re-encryption and symmetric encryption to encrypt attributes. This scheme combines the features of the former with the efficiency of the latter. The tuple $(a_i, \sigma_I(a_i))$ is encrypted with a symmetric cipher, whereas the symmetric content encryption key is encrypted, re-encrypted, and decrypted with a proxy re-encryption cipher.

5 User Identity Management App

In this section, we propose a mobile app for key generation and handling authorization decisions. First, we discuss why a mobile app is a suitable platform. Then, we explain implementation details including the app's structure, communication channels, interaction and cryptographic processes. Finally, we conclude this section describing how the UIM app improves usability, for example, by minimizing user interaction.

Platform Choice: As the re-encryption key is generated from the user's private key, this process has to be performed inside the user's trusted domain. We enable the management of the user's identity and especially the generation of re-encryption keys through a mobile app due to the following benefits: Firstly, the ubiquity of mobile phones eliminates the need for installing software (e.g. desktop application, browser extension) and distributing the user's private key across multiple stationary devices, from which the identity transaction is initiated. A dedicated server could also be used for re-encryption key generation, but costs for deployment and maintenance are only viable for large user groups (e.g. company employees) and not for end users. Secondly, the TEE offers better protection of the key material than software based storages of web applications.

Components: The UIM app has two parts, namely the client application operating in the rich OS and the key generator operating as trusted application in the Trusted Execution Environment (TEE). Security critical operations involving the user's private key are performed in the TEE, which is a processor feature enabled in most modern mobile devices. Regular operations such as user interaction to select scopes and grant permissions are performed in the rich OS platform. Once the user installs the UIM app and registers at the IdP, the app generates a key pair (sk_U, pk_U) for the user in the TEE. The key generation operation in the TEE processor is isolated from the rest of the system using memory and I/O protection mechanisms. In our design, the user's private key never leaves the TEE. Also, the user's identity attributes are collected during the registration process. The UIM app signs these attributes $\mathbf{a} = (a_1, \ldots, a_m)$, encrypts both attributes and signatures into $\mathbf{c}_U = (c_{U1}, \ldots, c_{Un})$, and deposits the encrypted attributes at the IdP.

Notification: When the IdP requires user consent for sharing information with the SP, the user receives a push notification on her mobile device. The UIM app verifies the $CERT_{SP}$ and displays the identity of the requesting SP and the requested attributes. With just one click the user can express her consent, and then the KG generates a re-encryption key. Since the user's private key is used for generating the re-encryption key, the KG runs in the TEE to generate the $rk_{U \rightarrow SP}$, thereby, making it impossible for untrusted applications or other attacks on the mobile phone to extract the private key. The UIM app generates the keys once in a mobile device which the user always has on her, and thereafter, the user can access services from any device.

User-friendliness: A major advantage offered by our solution is that it is not required to distribute and maintain private key copies on all devices that should be used in an identity management process. Key-generation and authorization is a one-time process. The user's selections are remembered and suggested for reuse in subsequent authentication requests. Convenient (biometric) authentication mechanisms can also be integrated through flexible frameworks such as FIDO and WebAuthn.

6 Discussion

In this section, we discuss our IDaaS solution with respect to end-to-end confidentiality, user-centricity, key security, usability, and ease of integration:

End-to-end Confidentiality: Our goal is to achieve end-to-end confidentiality for identity data when stored in public cloud. We are able to fulfill this objective by integrating proxy re-encryption with OpenID Connect, which prevents an honest but curious IdP from accessing identity data in plaintext. The user's data can still be shared with authorized SPs by re-encrypting the ciphertext using a re-encryption key $rk_{U \to SP}$. Thus, our IDaaS solution provides enhanced privacy for identity provisioning.

User-Centric Control: The UIM app gives users full control over the data sharing process, as explicit consent from the user is required during the authentication and authorization process to generate the re-encryption keys. The IdP controls only the re-encryption process. Unlike [NA14], where a representative entity generates re-encryption key for all the users it represents, our approach allows its users to generate re-encryption keys themselves. This empowers the users to choose and trust SPs individually. Since the re-encryption key depends on the user's private key, the IdP cannot share data without user consent. This makes our IDaaS solution user-centric by design.

Key Security: The UIM app generates and stores cryptographic keys within the user's mobile device. More precisely, the user's private key is stored in the mobile device's TEE and the re-encryption key generation based on this private key is also performed in this environment. As the user's private key cannot be extracted from the TEE, unauthorized apps or other attacks on the mobile device can also not obtain this key material.

Usability: In our solution, the UIM app is the central actor for key-generation, encryption, authentication and authorization. Users do not have to install any additional software or browser-plugin for each computer or devices she uses to access protected services. Besides, single-sign-on (SSO), the UIM app ensures minimal user interaction; information about the user's recent authentication is stored and can be reused to complete a subsequent authentication request without requiring user interaction. The user generates her cryptographic keys and re-encryption key for a particular service provider only once and thereafter, she can access the SP's protected resources from any device. In addition, the user can change her initial authorization decisions anytime using the UIM app inducing the letter to request the deletion of the generated re-encryption keys by the *IdP*.

Ease of Integration: To adopt our IDaaS solution, existing service providers only have to perform two steps: SP's have to register their certificates $CERT_{SP}$ at the IdP during dynamic registration and implement a decryption function for the re-encrypted attributes. In general, we designed our approach of integrating proxy re-encryption into OpenID Connect so that changes to existing OpenID Connect deployment are kept to a minimum. We used established protocols and stayed close to their existing configurations and setting. Hence, our approach enables a convenient integration into existing infrastructures and protocols.

Limitation: While our contribution focused on confidentiality aspects in identity management, we do not provide unlinkability and anonymity between federated identity provider and service providers. Hence, it is evident to mention that the proposed IDaaS tackles the privacy concerns related to data curious behavior of the IdP, whereas preventing the IdP to profile the user's behavior across services is beyond the scope of this paper.

7 Conclusion and Future Work

We proposed a privacy-preserving, user-centric and user-friendly identity management solution consisting of two parts: Firstly, we integrated proxy re-encryption into OpenID Connect to ensure end-to-end data confidentiality for identity provisioning in both web-based and mobile applications. When a service provider requires identity data, these selected attributes are re-encrypted upon user consent, and by design only the service provider can decrypt the attributes. Users stay in control as they have to generate a re-encryption key to enable the data sharing process. Also, this key generation only has to happen once per service provider, so the solution requires less user interaction. Secondly, to concretely implement our vision, we introduced a User Identity Management (UIM) App for nowadays omnipresent smart phones which takes the user's perspective into account. This app represents a trusted domain to conveniently perform the cryptographic operations on the user's side. The user only has to install this app once on her phone which she typically always has with her and does not have to install additional software to provision key material for each computer, tablet or other device she is using to initiate the transactions. This app also employs the TEE which is becoming increasingly available on modern phones to store the user's private keys while being flexible enough to support the integration of arbitrary cryptographic schemes needed to implement proxy re-encryption. As the user's private key never leaves the TEE, it is also more secure against attacks on the phone.

For future development, we plan to add a key recovery functionality in our proposed system, where users will have the capability to recover key material when necessary in a secure and trusted manner. Additionally, we plan to implement our approach within the e-Health, e-Business, and e-Government pilots specified by the CREDENTIAL[3] project. We also plan to integrate proxy re-encryption with redactable signatures [Jo02] to achieve selective disclosure on parts of a signed document. Furthermore, we plan to work on integrating conditional proxy re-encryption [We09], which would further reduce the trust required in the identity provider by limiting for which data a re-encryption key can be (mis-)used.

Acknowledgments

This research was supported by the CREDENTIAL project, which received funding by the European Union's Horizon 2020 research and innovation programme under grant agreement No 653454.

[3] CREDENTIAL: Secure Cloud Identity Wallet, https://credential.eu/

References

[BBS98] Blaze, M.; Bleumer, G.; Strauss, M.: Divertible Protocols and Atomic Proxy Cryptography. In: Advances in Cryptology — EUROCRYPT'98, volume 1403 of Lecture Notes in Computer Science, pp. 127–144. 1998.

[Ca09] Cantor, S.; Kemp, J.; Philpott, R.; Maler, E.: Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0 - Errata Composite. 2009.

[CS10] Chen, Y.; Sion, R.: On Securing Untrusted Clouds with Cryptography. In: Proceedings of the 9th Annual Workshop on Privacy in the Electronic Society. ACM, pp. 109–114, 2010.

[GN09] Goodner, M.; Nadalin, A.: Web Services Federation Language (WS-Federation) Version 1.2. 2009.

[Hö16] Hörandner, F.; Krenn, S.; Migliavacca, A.; Thiemer, F.; Zwattendorfer, B.: CREDENTIAL: A Framework for Privacy-Preserving Cloud-Based Data Sharing. In: 11th International Conference on Availability, Reliability and Security. 2016.

[Jo02] Johnson, R.; Molnar, D.; Song, D. Xiaodong; Wagner, D.: Homomorphic Signature Schemes. In: CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA. pp. 244–262, 2002.

[LZ10] Leitold, H.; Zwattendorfer, B.: STORK: Architecture, Implementation and Pilots. In: ISSE 2010 - Securing Electronic Business Processes, Highlights of the Information Security Solutions Europe Conference. Vieweg+Teubner, pp. 131–142, 2010.

[NA14] Nuñez, D.; Agudo, I.: BlindIdM: A Privacy-Preserving Approach for Identity Management as a Service. International Journal of Information Security, 13(2):199–215, 2014.

[NAL12] Nuñez, D.; Agudo, I.; Lopez, J.: Integrating OpenID with Proxy Re-Encryption to Enhance Privacy in Cloud-Based Identity Services. In: 2012 IEEE 4th International Conference on Cloud Computing Technology and Science. IEEE, pp. 241–248, 2012.

[Sa14] Sakimura, N.; Bradley, J.; Jones, M.; de Medeiros, B.; Mortimore, C.: OpenID Connect Core 1.0. 2014.

[SBJ11] Sakimura, N.; Bradley, J.; Jones, M.: OpenID Connect Dynamic Client Registration 1.0. 2011.

[We09] Weng, J.; Deng, R. H.; Ding, X.; Chu, C.; Lai, J.: Conditional Proxy Re-Encryption Secure Against Chosen-Ciphertext Attack. In: ACM Symposium on Information, Computer and Communications Security, ASIACCS 2009, Sydney, Australia. pp. 322–332, 2009.

[ZS15] Zwattendorfer, B.; Slamanig, D.: Design Strategies for a Privacy-Friendly Austrian eID System in the Public Cloud. Computers & Security, 52:178–193, 2015.

[ZS16] Zwattendorfer, B.; Slamanig, D.: The Austrian eID Ecosystem in the Public Cloud: How to Obtain Privacy while Preserving Practicality. Journal of Information Security and Applications, 27-28:35–53, 2016.

[Zw14] Zwattendorfer, B.; Slamanig, D.; Stranacher, K.; Hörandner, F.: A Federated Cloud Identity Broker-Model for Enhanced Privacy via Proxy Re-Encryption. In: IFIP International Conference on Communications and Multimedia Security. Springer, pp. 92–103, 2014.