



HAL
open science

Identity management in cross-cloud environments: Towards self-sovereign identities using current solutions

Mohamed-Amine Ben Haj Salah, Romain Laborde, Abdelmalek Benzekri,
Mohamed Ali Kandi, Afonso Ferreira

► **To cite this version:**

Mohamed-Amine Ben Haj Salah, Romain Laborde, Abdelmalek Benzekri, Mohamed Ali Kandi, Afonso Ferreira. Identity management in cross-cloud environments: Towards self-sovereign identities using current solutions. 2024. hal-04668399

HAL Id: hal-04668399

<https://hal.science/hal-04668399v1>

Preprint submitted on 6 Aug 2024

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

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

Identity management in cross-cloud environments: Towards self-sovereign identities using current solutions

Mohamed Amine Ben Haj Salah, Romain Laborde, Abdelmalek Benzekri,
Mohamed Ali Kandi, and Afonso Ferreira

IRIT, Université de Toulouse, CNRS, Toulouse INP, UT3, Toulouse, France
Mohamed.Ben-Haj-Salah@irit.fr

Abstract. Cross-cloud is a strategy followed by organisations to use services provided by multiple cloud service providers. To provide a seamless experience, IAM interoperability issues need to be addressed. Current approaches involve relying on a third party cloud broker which causes known-problems that originally led to the concept of cross-cloud. In this article, we analyze the IAM interoperability issues when building a cross-cloud environment on top of the current major cloud service providers (Amazon Web Service, Microsoft Azure and Google Cloud Platform) by implementing a cross-cloud storage application. This experiment demonstrates the need to integrate new identity management systems to cloud IAM services in the future. We then propose a cross-cloud identity management model that follows the Self-Sovereign Identities principles and which is interoperable with existing cloud IAM services.

Keywords: Identity and access management · cross-cloud · cloud security · self-sovereign identities

1 Introduction

Cross-cloud is a strategy followed by organisations to use services provided by multiple cloud service providers (CSP). By combining multiple solutions from different CSPs, cloud architects can select the best features available in the different cloud service-level agreements (SLA) to build advanced cloud based services. Some of the major issues addressed by a cross-cloud environment are:

- **Scalability** : A part from the cloud native feature of scalability, cross-cloud environments enable organisations to deploy resources closer to end-users, reducing latency, improving performance and expanding the architecture easily, fastly and efficiently.
- **Following constraints and laws** : Organisations can deploy resources in geographically distributed data centers to comply with with residency regulations (e.g., GDPR [14], HIPAA[16] or EUCS[15]) and choose CSPs that offer services and compliance certifications aligned with regulatory requirements.

- **Avoiding vendor lock-in and cost efficiency** : Organisations can mitigate the risk of dependency on single vendor lock-in and maintain negotiating power by seamlessly moving data and workloads from one CSP to another in a cross-cloud architecture. This allows them to change CSP or redistribute workloads based on evolving requirements, pricing models, or performance guarantees. Thus, organisations can respond quickly to changes in service offerings and SLAs.
- **Disasters avoidance and recovery** : To ensure data resilience and facilitate rapid recovery in the event of data loss or corruption, cross-cloud architectures enable and promote data replication and backup in multiple geographic locations. In addition, traffic can be rerouted to alternative providers without interruption. This redundancy minimizes the downtime risk and ensures continuous availability, especially during peak demand periods.
- **Selecting and combining different services for their unique services not provided elsewhere** : By selecting the most appropriate services from different providers, organisations can access a wide range of capabilities not available elsewhere. For example, they can use AWS for its extensive AI and machine learning tools, GCP for its advanced data analytics services and Azure for its robust enterprise-grade applications. Furthermore, by combining multiple services, organisations can create new services tailored to their needs.

However, CSPs implement different Identity and Access Management (IAM) approaches which limits multiple cloud seamless integration. This can lead to some incompatibility and limitations when building a cross-cloud environment due to the restrictions imposed by each CSP. Indeed, some CSPs provide solutions to pipeline workloads by offering services that manages identity federations between different CSP. Nevertheless, these solutions tend to be strict in their configuration and don't give the administrator the freedom to use a unique identity across all the CSP in a cross-cloud environment, at least without using a CSP that manages these federations.

We present in this article a practical study that experiments the implementation of a cross-cloud storage environment built on top of three major CSP services, namely Amazon AWS S3, Microsoft Azure and Google Cloud Storage. We studied the different IAM services provided by each CSP and implemented two different strategies to enable our cross-cloud storage application to slice files into blocks that are stored in multiple clouds. This practical experience enables us to discuss IAM interoperability issues that need to be addressed to provide a seamless experience across the cross-cloud environment. As a consequence, we present a user-centric and cross-cloud Identity model compliant with principles proposed by the Self Sovereign Identities which aim at giving full control to users on their digital identities. The advantage of our solution is that it can be implemented using existing cloud providers solutions.

The rest of the article is structured as follows. In section 2, we clarify the definition of cross-cloud. Then, in section 3 we present the related work on the identity and access management in a cross-cloud environment. Section 4 describes the practical study that we implemented and presents its results. In section 5, we

present a model capable of identifying users based on the Self-Sovereign Identity principles. Finally, we conclude and indicate the future work in section 6.

2 What is cross-cloud?

Unlike the concept of cloud computing which has been clearly defined by the NIST [1], cross-cloud has no such standard definition. In addition, other related terms such as multi-cloud are used in the literature. Sometimes these two terms are used interchangeably, leading to confusion. In this section, we list the different interpretations of the cross-cloud and the multi-cloud concepts and present the definition that will be considered in the rest of this article.

CSPs such as Cloudflare define multi-cloud as follows. "Multi-cloud means multiple public clouds. A company that uses a multi-cloud deployment incorporates multiple public clouds from more than one cloud provider." [8].

Some sources present the concept of cross-cloud either in a similar or a different way from multi-cloud. On the one hand, the differences between cross-cloud and multi-cloud are defined by [5] in this way "Unlike multi-cloud, which describes an environment that uses multiple cloud service providers, cross-cloud refers to an application or workload that uses multiple cloud service providers (CSPs) and transfers data seamlessly across those CSPs." This idea is shared by [3] and [2] for whom cross-cloud favours the flow of data across multiple public clouds whereas multi-cloud creates data silos and borders between clouds that trap the data and force the user to work on copies. To add to the confusion VMware [4] has a product called *VMware cross-cloud services* presented as follows "The VMware Cross-Cloud services portfolio provides multi-cloud services for all apps"[4]. So here, VMware uses the term multi-cloud to describe an environment with multiple cloud instances and uses the term cross-cloud to name their product.

Elkhatib [6] defines cross-cloud applications as applications that consume more than one cloud API under a single version of the application. According to Elkhatib, multi-cloud is a sub type of cross-cloud systems. Here a multi-cloud system is defined by its provisioning means which is -according to the author- a least common denominator API via a common programming model or a translation library. Along with some other specifications, the author implicitly defines multi-cloud as an evolution from hybrid cloud (combining private and public cloud) that brings some abstraction (to the user and developer) but may lose some specific features if not correctly configured.

Finally, Hong et al. [7] pointed out the confusion in the definitions out there. They compared research articles to mention the inconsistency and the confusion when authors and researchers define multi-cloud and cross-cloud. Unlike [6], the author categorises multi-cloud and cross-cloud as two distinct types of cloud. Hong et al. state that multi-cloud is better suited to users that utilize different cloud services for different applications in their business and the cross-cloud as designed to make the transfer of data and utilisation of apps across the clouds more streamlined and cohesive. Furthermore, the authors mentioned that a cross-cloud application uses more than one cloud API under a single version of the application and

that in a multi-cloud environment applications are hosted as chunks among an heterogeneous network of different clouds.

We can thus see that there is no real agreement on the definition of cross-cloud. Therefore, for the rest of the paper, we will consider that a cross-cloud environment is composed of multiple cloud systems controlled by different public cloud providers (with at least one public cloud provider) that are combined for different roles and workloads guaranteeing a seamless experience.

3 Related work

Although the importance of the problem, only few researchers have investigated the IAM interoperability issues in the context of multi-cloud environments.

Zahoor *et al.* [10] point out the inconsistency and conflicts between IAM policies in multi-cloud environments. Indeed, most CSPs are providing proprietary cloud solutions and have their own IAM policies and processes that can be designed on different models such as Role Based Access Control, Attribute Based Access Control or User Based Access Control. The solution presented by Zahoor *et al.* is based on the Attribute Based Access Control model to aggregate different (possibly conflicting) authorization policies expressed in different access control models or different implementations of the same model. This solution resolves conflicts between the different policies provided by the CSPs and helps with the governance of a multi-cloud environment. However, this article only considers access control policies and doesn't investigate the identity interoperability issue in such an environment.

Some related works propose to tackle the multi-cloud interoperability issue by introducing Cloud Service Brokers. A Cloud Service Broker (CSB) is defined as a "middleman between the buyer and sellers of cloud computing services, serving as a middle layer in this process" [9]. There are two types of CSBs : cloud aggregators and cloud customizers. A cloud aggregator offers all services provided by the CSPs through one single interface. On the other hand, the main goal of cloud customizers is to provide new services that are composed of different other services provided by the CSPs. CSB can negotiate with CSP on behalf of the customer making them an extra stakeholder in a multi-cloud environment. However, the recent state of the art produced by burak [9] shows that CSB don't manage IAM in a multi-cloud environment.

Mulder [12] presents a multi-cloud administration guide where he presents two possible solutions to handle IAM interoperability issue: the Identity as a Service (IDaaS) and the Cloud Access Security Broker (CASB). According to the author, IDaaS "provides a centralized platform for managing user identities, access controls, and authentication across multiple cloud services and applications", while CASB "provides a layer of security and access control between cloud services and end-users". The main difference between these two solutions lies in that CASBs are usually used for monitoring user access to cloud services, while IDaaS is used for synchronizing user directories, syncing password hashes and authentication users to active directory federation services [13].

Sukmana et al. [17] propose a unified cloud access control model to integrate into cloud storage brokers. This model provides IAM and storage functions including identity provisioning, authentication and authorisation. This work has been implemented on AWS and GCP only because of the similarities in their storage and IAM services. A user's identity here is comprised of 2 "sub identities" : *User* for AWS and *Service Account* for GCP which are created on demand. The heart of this model is the Cloud Access Control Policy (CACP) which is an entitlement matrix and Access Control List by mapping access information (identity, access keys ...) with a set of privileges or allowed actions in the CSP. The CACP manages the storage resources access.

Previous researches have investigated different strategies to solve the IAM interoperability issue. The policy aggregator proposed by [10] is limited to access control heterogeneity only. It doesn't address identity management. The other approaches introduce a broker entity. However, this strategy only postpone the initial issues that led to multi-cloud solutions because customers can be locked into the broker's environment. If a broker decides to drop a CSP from his catalogue, the customer organisations should be either constrained by this list of CSPs or forced to change to another broker.

4 A practical experiment using current cloud solutions

Our objective is to analyze if it is currently possible to unify the identity management in a cross-cloud environment without the need of a broker. We want to evaluate if currently available technologies allow a cross-cloud identity to seamlessly log on to all the CSPs. As a use case scenario, we developed a basic cross-cloud storage application which allows to distribute files among different CSPs. We chose to build our cross-cloud application on top of Amazon Web Services (AWS), Microsoft Azure and Google Cloud Platform (GCP). AWS, Azure and GCP are the three major CSPs that represent respectively 31%, 25% and 11% of the public cloud market share in Q1 2024¹. This section describes the IAM solutions provided by the three CSPs and the different implementations we tested to produce a seamless unified identity system for our cross-cloud storage application.

4.1 Identity and access management in AWS, Azure and GCP

Each company designs and develops its product differently without necessarily prioritising interoperability with the other cloud solutions. Therefore, the three cloud solutions follow different IAM approaches (Figure 1).

First, AWS and Azure support two access control models namely Attribute-based access control (ABAC) and Role-based access control (RBAC). However, GCP only implements RBAC which limits the expression capabilities compared to AWS and Azure.

¹ <https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/>

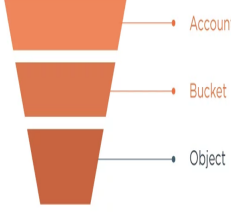
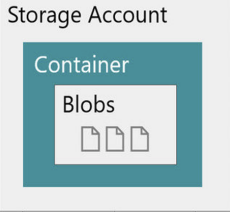

The structure of the policies is also different. AWS combines permissions and resources into one JSON file. The pairs (permission, resource) for accessing resources are associated with an identity or a role through a policy in another JSON file. On the other hand, in Azure and GCP, the philosophy is to separate the permissions from the scope (i.e. a set of resources). Permissions are attached to a role and users are assigned to a pair of role and scope. Consequently, AWS attaches a policy (permissions + resources) to a role, while GCP and Azure assign a role (set of permissions) and a set of resources to a user.

Cross-cloud IAM interoperability is also affected by how each CSP structures resources within its cloud solution. AWS and GCP have similar structure with *buckets* acting as a repository containing objects. On the other hand, Azure specifies storage account that includes containers holding blobs (files). These differences in structure are described in table 1. The format of the identifiers Identifier is also not standardized. AWS uses the amazon resource format which is a kind of a uniform resource name identifier while identifiers in Azure and GCP are uniform resource location.

Furthermore, each CSP also has its own methods for providing external access. The first approach is to manage local accounts using the IAM service of the CSP. In AWS, administrators can establish a resource-based policy wherein an IAM role is created and configured specifically (usually with lesser permissions than the original role) to allow external assumption by workflows or applications (usually done with certificates). In GCP, the process involves creating a binding for an external identity onto a service account, granted with a role susceptible to impersonation. The workflow/application then uses a *service account key* (JSON file generated by GCP containing access information) in order to impersonate the role. Azure, on the other hand, provides account access keys for the storage account created simplifying its access. In conclusion, each cloud platform adopts distinct approaches to enable external access and identity management, tailored to their respective architectures and security paradigms.

Another approach to provide external access to entities consists in outsourcing the management of users accounts to Identity Providers. These methods generally point to federated identities solutions and are part of the IAM services provided by the CSP like Cognito (AWS), Entra ID (Azure) and Workload Identity Federation (GCP). While all three services focus on identity management, they differ in their primary use cases and integration capabilities. *Azure Entra ID* is well-integrated with Microsoft's ecosystem and caters to enterprise needs while *Cognito* is tailored to manage identity federations for authentication and authorisation, and *Workload Identity Federation* in GCP focuses on securing access control for workloads while leveraging existing identity infrastructure. Finally, the difference also lies in the degree of freedom granted to administrators. While AWS and GCP allow an administrator to configure any IdP to outsource users accounts management, this is not possible with Azure, which imposes the use of either an Azure Active Directory, or an IdP from the list of IdPs verified by Microsoft.

Table 1. Overview of IAM and storage specifics of AWS, Azure and GCP

	AWS	Azure	GCP
Access control model implemented	RBAC/ABAC	RBAC/ABAC	RBAC
Policy structure	Permissions to resources are assigned to an identity via a policy	Permissions are attached to a role. User is assigned a pair of role and scope (set of resources)	Permissions are attached to a role. User is assigned a pair of role and scope (set of resources)
Resource structure			
Resource identifier format	Amazon resource name (e.g. <code>arn:aws:sqs:us-east-1:555555555555:teste-43523523543</code>)	Uniform Resource Identifier (e.g. <code>https://myaccount.blob.core.windows.net/mycontainer/myblob</code>)	Google Cloud Platform blob identifier (e.g. <code>gs://bucket_name/object_name</code>)
Locally managed accounts	Standard X.509 Certificate	Account key	Service account key
Externally managed account	Service : <i>Cognito</i> Identity federation with external IdP (any third-party IdP accepted) applicable for human users and workflows	Service : <i>Entra ID</i> Identity federation with limited external verified IdP (e.g. Facebook) or another verified CSP by Microsoft	Service : <i>Workload Identity Federation</i> Identity federation with external IdP (any third-party IdP accepted) for workflows only

4.2 Experimentation

The aim of this section is to evaluate the different possibilities offered by the main CSPs available on the market for managing multiple identities of a cross-cloud application. The goal is to achieve a seamless and unified IAM integration in the cross-cloud environment. As a use-case example, we have developed a cross-cloud storage application that slices files into blocks that are distributed across multiple CSPs. The implemented scenario is illustrated in figure 1 where a pdf file divided into 9 blocks and each CSP (AWS, Azure and GCP) holds 3, then the script tries to log in to the 3 CSPs and download the 9 parts before assembling them to create

a single pdf file again. The source code of the implementations is available on github².

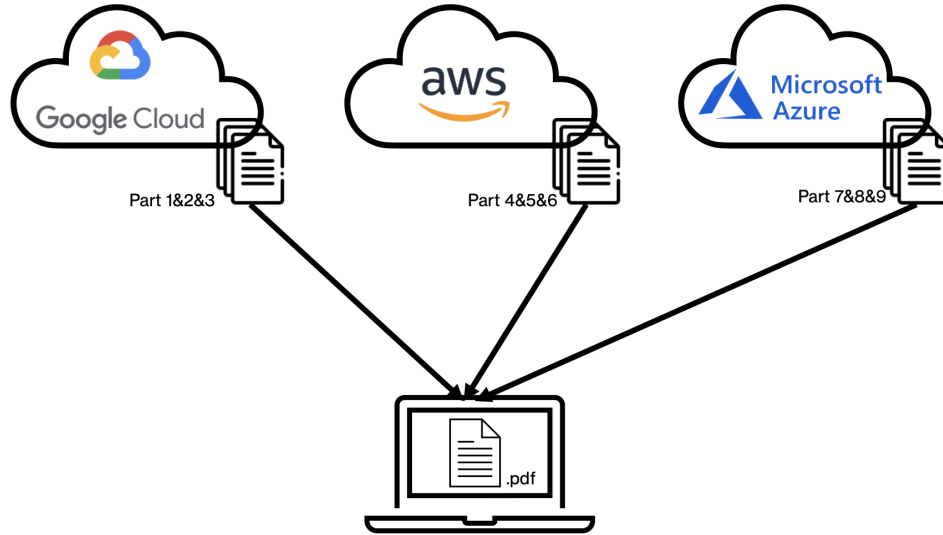


Fig. 1. Use case scenario

Locally managed accounts The first experimentation takes advantage of the access credentials provided by the CSP for external use (usually reserved for workflows and applications) using locally managed accounts. The process starts by setting up the adequate roles and accounts for each CSP. An administrator needs to create a different *profile* in each CSP. These profiles will be assigned to a workload in AWS, to a service account specific for workloads in GCP and to the storage account in Azure. The next step is the retrieval of the access credentials from each CSP listed in the figure below 2. In order to authenticate with the CSPs, the workflow needs to provide the adequate certificate and private key for AWS, the service account key which is a Json file containing specific credentials (private key, client id, x509 certificate, ...) for GCP and the 512-bit storage account symmetric key for Azure.

To establish the exchange, the app uses the libraries provided by the CSPs : boto3 for AWS, azure.storage.blob for Azure and google.cloud for GCP. These are the storage libraries that are providing the necessary functions to use in order to authenticate the app and then to look up and download the wished file.

In order to authenticate the app, these libraries need some prior configuration to use the correct credentials. These configurations are different from a CSP to another. In AWS, the admin needs to download the *AWS command line interface* and set up a profile on the local machine with the correct credentials. This profile

² <https://github.com/amineIRIT/cross-cloud-downloader>

will then be called by the authentication function from the library. For Google cloud, the set up happens in the application's python script where the admin needs to assign the Json file's (containing the key) path to the environment variable `GOOGLE_APPLICATION_CREDENTIALS`. This variable will then be fetched by the authentication call. On the other hand, no pre-configuration is needed for Azure, the authentication function takes the account name and the symmetric key as parameters.

After the authentication phase, as the accounts are locally managed, the workflow simply assumes (through the standard IAM process) the roles created earlier that gives it the read only rights, sufficient to look up and downloads the requested files.

The issue here is that the script uses a different identity (user profile) for each CSP and judging by the quantity and diversity of information needed for authentication (cf. table 2), it is neither a scalable nor efficient solution.

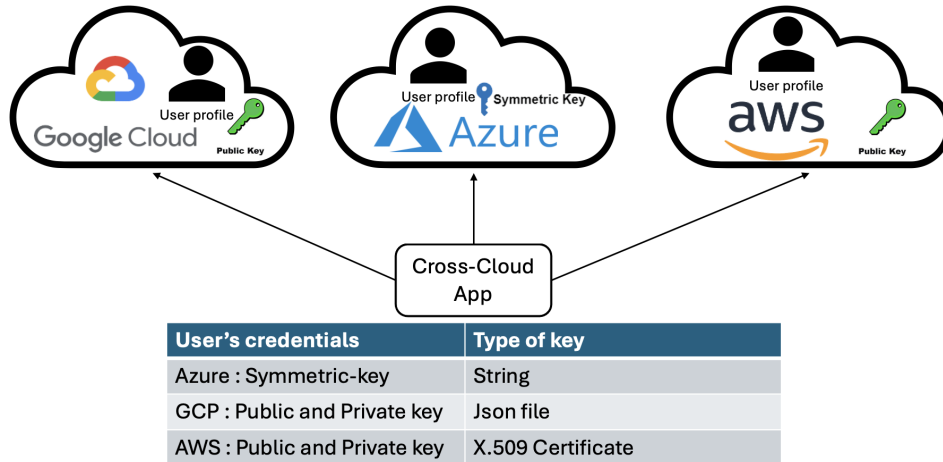


Fig. 2. High level view of the exchanges between the application and the CSPs

Externally managed account For this second experimentation, our goal is to have control over the authentication process in order to implement the unique identity management solution. Therefore, we used Open Id Connect (OIDC) which is an authentication protocol based on OAuth 2.0 that allows the verification of user identities via an authorization server and issue tokens to prove the success of the authentication. It is open source and maintained by the OpenId Foundation. Therefore, there are plenty of solution and software to set up a OIDC IdP instance. For our study case, we chose *Keycloak*³, an open-source identity management tool. The use case's new architecture is presented in 3.

³ <https://www.keycloak.org/>

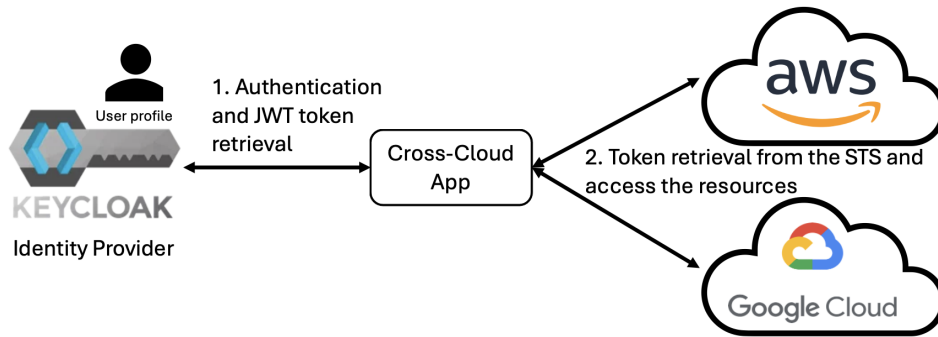


Fig. 3. High level view of the architecture linking the application, the OIDC identity provider and the CSPs

We started by deploying an online instance of Keycloak to implement an OIDC IdP. In order to authenticate users, the IdP should know the clients, i.e. the entities representing the application or web site that will consume the OIDC tokens generated by Keycloak. For this practical study, we configured similar clients on Keycloak for each CSPs to authenticate users with the Authorization flow and Proof Key for Code Exchange (PKCE). PKCE is an extension to the Authorization Code flow to prevent Cross-Site Request Forgery (CSRF) and authorization code injection attacks⁴.

Then, an administrator needs to configure the CSPs to trust the identity provider in order to accept the access token later on. This is where we found the first interoperability issue since Azure only support SAML identity federations with third party IdPs that are not included in the Microsoft's trust list. On the other hand, AWS and GCP provide two key services: AWS cognito (*identity pools*) and Workload Identity Federation for GCP. These two services are essential because they are responsible for the exchange of the OIDC token for Google Cloud or AWS credentials. Setting up the trust relationship between AWS or GCP and our IdP was straightforward: the needed information by the CSPs is the provider's endpoint, a thumbprint (hash value of the server's X.509 certificate) and the audience (the OIDC Client ID created earlier).

Next, the administrator needs to assign a set of permissions to the users authenticating from the provider. This is done by assigning them roles to claim after the authentication with the CSP, either by granting users a default role based on the audience in the OIDC access token or a role according to mapping rules based on claims (attributes) included in the access token. In our use case, we only set up one specific role that have read rights on the storage buckets. This will be the default role assigned to the users that pass the authentication process (present a valid access token).

Finally, the users accounts are created on the IdP to make the identities to establish and activate the identities in our cross-cloud environment.

⁴ <https://oauth.net/2/pkce/>

In the authorization grant flow with PKCE, the process starts when the user tries to log in to a client application. It generates a unique code verifier and transforms it into a code challenge which is then sent to the authorization server along with the user’s login request. Next, the user is redirected to authenticate on a login page usually provided by the authorization server. After a successful authentication, the server returns an authorization code to the client app which will send back this authorization code along with the original code verifier to the authorization server. The server validates the code verifier against the code challenge it received earlier. Afterward, if the validation is successful, the server issues tokens (ID token and access token) to the client. These tokens will allow the client app to securely access protected resources on behalf of the user. In our case the user is the cross-cloud app and the client application is the CSP storage service. Giving that a script can’t interact with redirected links and to simplify the implementation, we had to configure the cross-cloud app to impersonate the cloud service in order to successfully pass the PKCE verification.

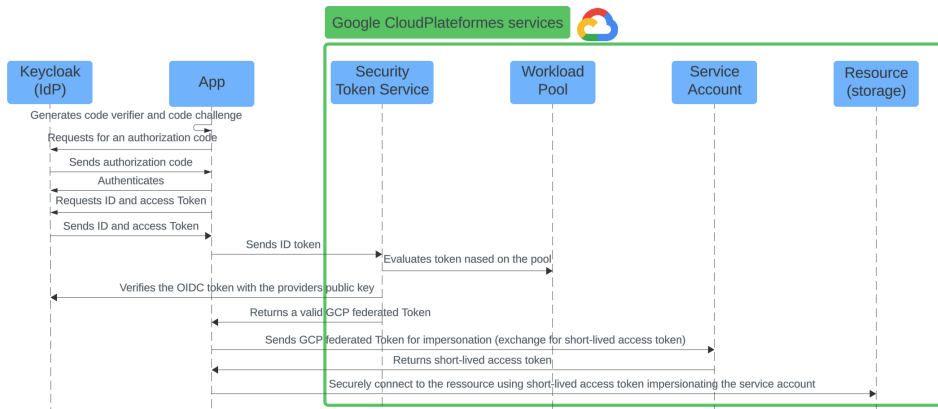


Fig. 4. Exchanges between the application, the OIDC identity provider and Google Cloud services

Figure 4 represents the exchanges between the python script implementing the cross-cloud storage application, The OIDC identity provider and the various services of Google Cloud. To simplify, this figure does not include the exchanges with the AWS services which are significantly similar.

To start, the script generates a code verifier and code challenge for the PKCE flow. Then, it requests and retrieves an authorization code from identity provider (Keycloak). After that, the app sends a login request with the user’s credentials and extracts the authorization code from the response. Finally, it sends a request to the IdP to obtain an access token using the authorization code.

After obtaining the access token, the app uses it to authenticate with google cloud services by forwarding the IdP token to GCP’s Security Token Service (STS), which verifies the token’s audience and issuer using public keys published by the

OIDC provider. After validation, the STS issues a short-lived (1 hour) federated GCP token representing the Service Account credentials, allowing access to GCP resources.

The process for AWS is quite similar where an STS service is responsible for verifying a token and issuing temporary AWS credentials based on claims from an OIDC IdP. As explained previously, our experimentation can only support AWS and GCP due to limitations imposed by Microsoft where Azure only trusts certain OIDC identity provider (Facebook, Google, ...).

5 Towards cross-cloud and self-sovereign identities

The locally managed accounts strategy enables to build a cross-cloud application on using any CSPs. However, this solution is not scalable and requires administrators to manage and coordinate as many accounts as CSPs for each user. Moreover, this means combining as many authentication methods as there are CSPs.

The externally managed accounts approach based on OIDC allows the cross-cloud application to use a single identity for each user across the different clouds to gain access to the resources. Such an approach enables the organisation that wants to build a cross-cloud environment to use its own IdP to manage the identity of users. However, this solution comes with some interoperability issues and requires the identities to be managed by the cross-cloud broker.

This solution therefore requires the organization deploying a cross-cloud solution to manage the entire lifecycle of users identities [25], or to establish an identity federation as part of a collaboration with other organizations. This type of solution is therefore highly restrictive and very difficult to deploy when users are not under the control of the organization.

Identity federations also raises a number of issues related to the protection of personal information [22] and the independence of digital identities from applications, which is a major constraint in cross-cloud environments [23].

The Self-Sovereign Identity (SSI) concept aims to give people control of their identity information in the digital realm. Allen [26] defined the principles of self-sovereign identity as: access, existence, protection, consent, minimization, control, persistence, portability, interoperability, and transparency. Self-sovereign identity illustrates a new decentralized identity model where users are at the center and control the sharing of their identity.

There is an active community of researchers, organizations and companies working on this topic with numerous emerging new standardization efforts for exchanging identity attributes such as W3C Verifiable Credentials and new open protocols such as the OIDC for Verifiable Presentation (OIDC4VP [28]) and verifiable credential Issuance (OIDC4CI [29]), Self-Issued OpenID Provider (SIOP [27]) or proprietary systems (e.g., Microsoft Authenticator App and the IBM Verify). However, there is no mature implementation yet and none of these specifications has yet established itself as a standard.

Our objective is to propose a cross-cloud identity management system influenced by the SSI principles and built on top of currently available standard protocols. Figure 5 below presents the architecture of our cross-cloud identity solution.

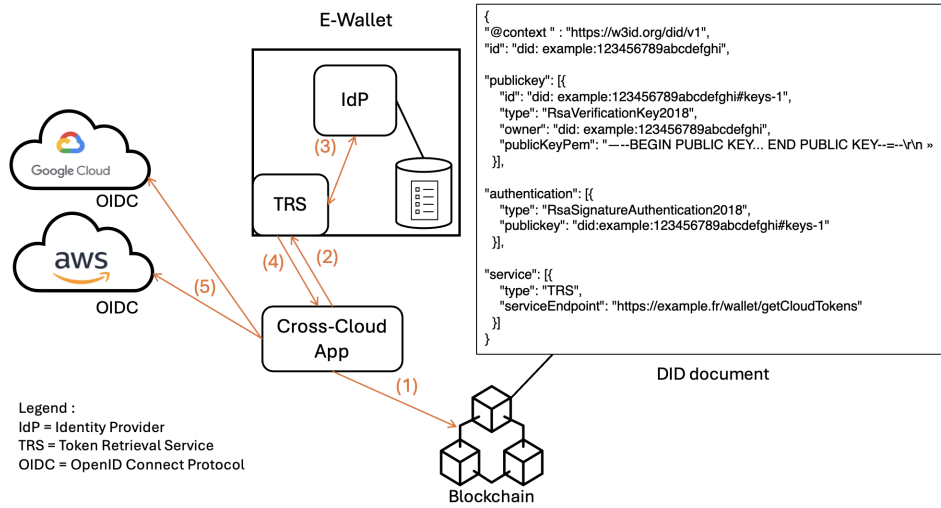


Fig. 5. SSI solution for a cross-cloud environment

In the previous solution implemented for our experiment described in section 4.2, the cross-cloud app was responsible for retrieving the IdP token, authenticating with the CSPs and downloading the wished file. Such an approach requires users to delegate the management of their credentials to the cross-cloud application in order to obtain the OIDC access tokens for each CSP. Conversely, our solution allows users to manage their identity and control which cross-application can access it. Therefore, we introduce an identity wallet controlled by users which consists of an IdP (e.g. Keycloak) and a new entity called Token Retrieval Service (TRS). The TRS module is a web service used by cross-applications to obtain identity tokens from the identity wallet while the IdP module is responsible for managing the identity attributes as well as controlling access of cross-cloud applications to the TRS.

When a user creates an account on a CSP, he configures the CSP identity management system to use the IdP of his identity wallet. This initialization step is repeated for each CSP that the user wants to use. The user needs also to create a new account on the IdP module for each cross-cloud application he wants to use and share the cross-cloud application credentials to allow it to authenticate on the IdP module of his wallet.

The cross cloud applications will require to know the URL of the TRS to obtain identity tokens. We propose to use W3C DID technology to simplify this task while preserving SSI principles. This new type of identifier is controlled and created by individuals and lasts for as long as their controller wishes to use it in a decentralized registry. A DID is simply a URI which resolves to a DID document that contains the key material and other metadata to reference services relevant to interactions with the DID subject. A DID is composed of three parts separated by ":" such as *did:scheme:identifier*. The *scheme* refers to the technology-specific verifiable data

registry used for recording DID documents, which can be any form of trusted data storage (e.g., distributed ledgers, decentralized file systems, databases of any kind, or peer-to-peer networks etc). The user can then easily publish his public keys and his identity wallet by adding the URL of the wallet in his DID document.

When the user launches the cross-cloud application, he provides his DID. The cross-cloud application first fetches the token retrieval service endpoint from the user's DID document (1). The cross-cloud application can then call the TRS specifying the list CSP for which it wants to obtain the identity tokens (2). After that, TRS starts authenticates the cross-cloud application using OIDC with the wallet IdP module. After authentication and authorization verification, the TRS starts an OIDC authorization grant flow with PKCE with the wallet IdP for each CSP in order to retrieve the CSP identity tokens (3). The TRS can then merge the different OIDC access token into a JSON file and return it to the cross-cloud application (4). Finally, the application can authenticate itself as the user at each CSP in order to perform its task (in our scenario, adding or retrieving pieces of file) (5).

Our solution respects most of Allen's SSI principles although built on top of currently and widely deployed protocols/standards. Indeed, our solution gives full control to users on which applications can obtain identity tokens, it is transparent and portable. Users' consent is required and applied by the user himself because he controls his identity wallet. And the identity wallet being managed by the users allows identities to be persistent. However, our solution is currently only compatible with the OIDC protocol. We need to extend it to implement SAML authentication.

6 Conclusion

Cross-cloud is an emerging cloud approach that allows organizations to combine different services from different cloud service providers to build advanced cloud based services. However, IAM interoperability issues need to be addressed to provide a seamless experience where multiple cloud service providers act as one.

In this article, we have evaluated the possibilities offered by the IAM services offered by the main CSPs available on the market to implement a cross-cloud application. We have shown that the two possible strategies are not satisfactory. We proposed an alternative solution based on SSI principles which can be implemented using current available identity management technologies.

This article is a preliminary assessment of the current constraints on implementing a cross-cloud system without using a broker. Our future work will focus on integrating the eIDAS EDIW into the cross-cloud identity management process. The European Parliament approved on 29 February 2024 the amendment of electronic IDentification, Authentication and trust Services (eIDAS⁵), which is a European Union regulation that provides a standardized framework for secure electronic interactions between citizens, businesses, and public authorities. The

⁵ <https://ec.europa.eu/digital-building-blocks/sites/display/EUDIGITALIDENTITYWALLET/EU+Digital+Identity+Wallet+Home>

new revision introduces a new EU Digital Identity Wallet (EDIW) and the related ecosystem to allow every person in Europe to exchange identity information from trusted private and public sources [31]. EDIW will benefit to cross-cloud IAM by providing high assurance and interoperable identities. This new ecosystem will provide identities with a high level of assurance, while limiting the effort involved in identity administration by the organization proposing the cross-cloud application. Although eIDAS is not SSI [30], it will also enable a more open system where users of cross-cloud solutions could compose electronic identities from attributes coming from various issuers.

7 Acknowledgment

This work was supported in part by the France 2030 ANR project ANR-23-PECL-0009 TRUSTINCloudS.

References

1. Mell, P. & Grance, T. The NIST Definition of Cloud Computing. (2011,9), <https://doi.org/10.6028/NIST.SP.800-145>
2. Beaucourt, E. Data : le cross-cloud réussit où le multcloud échoue. *Forbes France*. (2022,2), <https://www.forbes.fr/technologie/data-le-cross-cloud-reussit-ou-le-multcloud-echoue/>
3. Degnan, C. Don't settle for multi-cloud. Aspire to cross-cloud - Snowflake Blog. *Snowflake*. (2021,10), <https://www.snowflake.com/blog/dont-settle-for-multi-cloud-aspire-to-cross-cloud/>
4. VMware what is Multi-Cloud?. *VMware.*, <https://www.vmware.com/topics/glossary/content/multi-cloud.html>
5. Virtana *Virtana.*, <https://www.virtana.com/glossary/what-is-cross-cloud/>
6. Elkhatib, Y. Mapping Cross-Cloud Systems: Challenges and Opportunities.
7. Hong, J., Dreibholz, T., Schenkel, J. & Hu, J. An Overview of Multi-cloud Computing. (2019,3)
8. Cloudflare What is multi-cloud?. , <https://www.cloudflare.com/learning/cloud/what-is-multicloud/>
9. Cinar, B. The Role of Cloud Service Brokers: Enhancing Security and Compliance in Multi-cloud Environments. *Journal Of Engineering Research And Reports*. **25**, 1-11 (2023,10)
10. Zahoor, E., Ikram, A., Akhtar, S. & Perrin, O. Authorization Policies Specification and Consistency Management within Multi-cloud Environments. (Springer International Publishing,2018)
11. Shanahan, M. The Event Calculus Explained. *Artificial Intelligence Today*. **1600** pp. 409-430 (1999), http://link.springer.com/10.1007/3-540-48317-9_17
12. Mulder, J. Multi-Cloud Administration Guide: Manage and optimize cloud resources across Azure, AWS, GCP, and Alibaba Cloud (English Edition). (BPB Publications,2023), <https://univ-scholarvox-com.gorgone.univ-toulouse.fr/catalog/book/docid/88946820?searchterm=multi-cloud>
13. Abayomi-Zannu, T. & Odun-Ayo, I. Cloud Identity Management – A Critical Analysis. *Hong Kong*. (2019)

14. European Union, E. General Data Protection Regulation (GDPR) – Legal Text. *General Data Protection Regulation (GDPR)*., <https://gdpr-info.eu/>
15. ENISA EUCS – Cloud Services Scheme. ENISA., <https://www.enisa.europa.eu/publications/eucs-cloud-service-scheme>
16. Rights (OCR), O. Health Information Privacy. (2021), <https://www.hhs.gov/hipaa/index.html>
17. Sukmana, M., Torkura, K., Graupner, H. & Meinel, C. (2019) Unified Cloud Access Control Model for Cloud Storage Broker.
18. W3c Decentralized Identifiers (DIDs) v1.0. , <https://www.w3.org/TR/did-core/>
19. Grüner, A., Mühle, A. & Meinel, C. Analyzing Interoperability and Portability Concepts for Self-Sovereign Identity. *2021 IEEE 20th International Conference On Trust, Security And Privacy In Computing And Communications (TrustCom)*. pp. 587-597 (2021,10), <https://ieeexplore.ieee.org/abstract/document/9724325>
20. Cameron, A. & Grewe, O. An Overview of the Digital Identity Lifecycle (v2). *IDPro Body Of Knowledge*. **1** (2022,2), <https://bok.idpro.org/article/id/31/>
21. 24760, I. ISO/IEC 24760-1:2011 - Information technology – Security techniques – A framework for identity management – Part 1: Terminology and concepts. (ISO/IEC,2011), http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=57914
22. Chadwick, D., Laborde, R., Oglaza, A., Venant, R., Wazan, S. & Nijjar, M. Improved identity management with verifiable credentials and fido. *IEEE Communications Standards Magazine*. **3**, 14-20 (2019)
23. Aruna, M., Hasan, M., Islam, S., Mohan, K., Sharan, P. & Hassan, R. Cloud to cloud data migration using self sovereign identity for 5G and beyond. *Cluster Computing*. **25**, 2317-2331 (2022)
24. Naik, N. & Jenkins, P. Your identity is yours: Take back control of your identity using GDPR compatible self-sovereign identity. *2020 7th International Conference On Behavioural And Social Computing (BESC)*. pp. 1-6 (2020)
25. ISO/IEC JTC 1, Information technology, Subcommittee SC 27, IT Security Techniques. ISO/IEC 24760-1:2019 IT Security and Privacy — A framework for identity management — Part 1: Terminology and concepts, <https://www.iso.org/standard/77582.html>
26. Allen, “The Path to Self-Sovereign Identity.” Available: <https://www.lifewithalacrity.com/article/the-path-to-self-sovereign-identity/>
27. K. Yasuda, M. Jones, T. Lodderstedt. Self-Issued OpenID Provider v2 - draft 13. (Nov 2023) Available: https://openid.net/specs/openid-connect-self-issued-v2-1_0.html
28. O. Terbu, T. Lodderstedt, K. Yasuda, T. Looker. OpenID for Verifiable Presentations - draft 20. (Nov 2023) https://openid.net/specs/openid-4-verifiable-presentations-1_0.html
29. T. Lodderstedt, K. Yasuda, T. Looker. OpenID for Verifiable Credential Issuance - draft 13. (Feb 2024). Available: https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html
30. Lepore, C., Laborde, R., Eynard, J., Kandi, M. A., Macilotti, G., Ferreira, A., Sibilla, M. (2024, March). A Model For Assessing The Adherence of E-Identity Solutions To Self-Sovereign Identity. In *World Conference on Information Systems and Technologies* (pp. 153-163). Cham: Springer Nature Switzerland.
31. C. Lepore, R. Laborde, J. Eynard. (2024). Aligning eIDAS and Trust Over IP: A Mapping Approach. 21th International Workshop on Trust, Privacy and Security in the Digital Society (Trustbus@ARES).