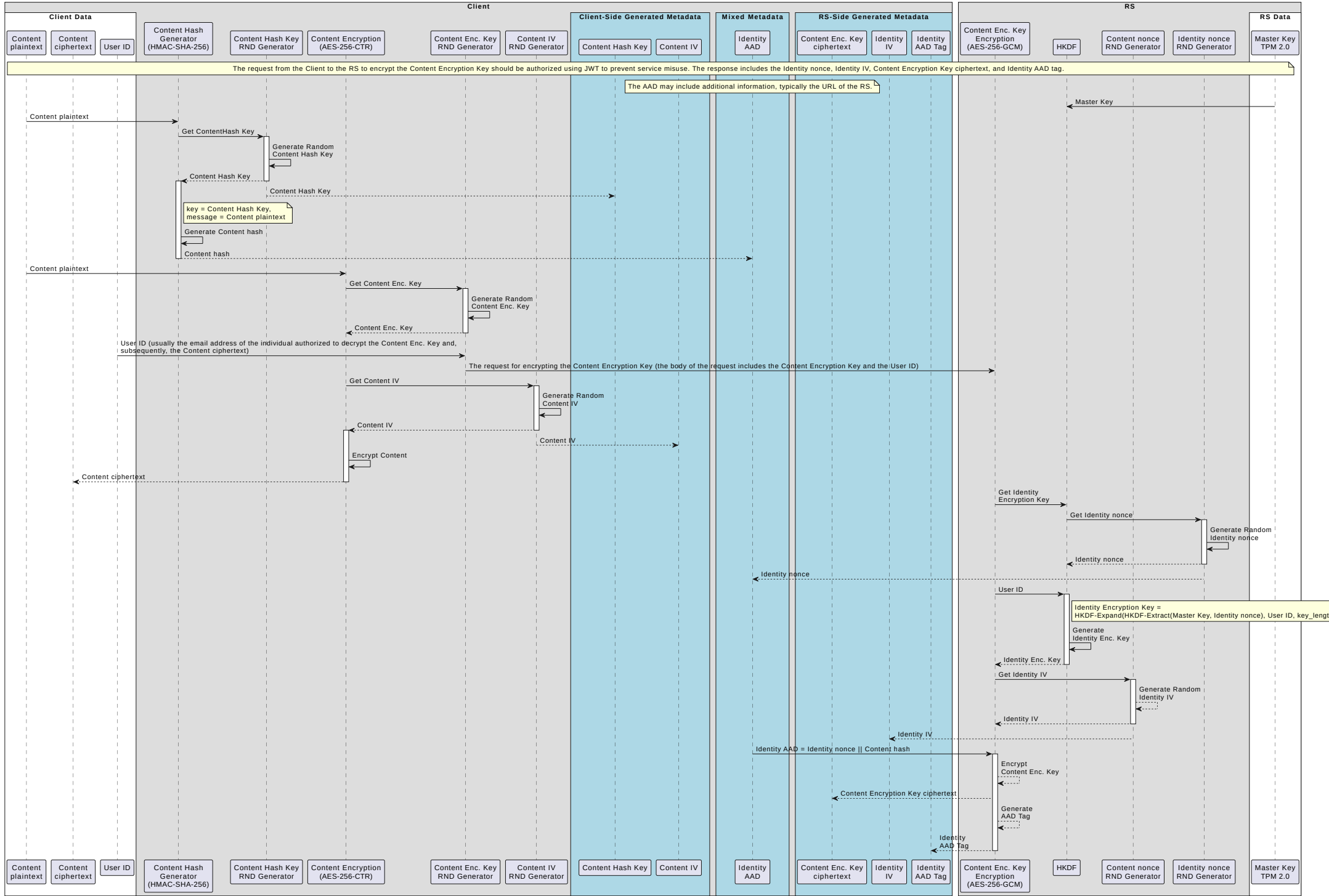# Identity-Based Symmetric Key Cryptography

Introduction

This is a set of Identity-Based Symmetric Key Cryptography Mechanisms. The client generates a content encryption key to encrypt the user's content. This key is then sent to the Resource Server for further encryption, along with the content hash and the user's ID (email address) of the individual authorized to decrypt it. The Resource Server returns the encrypted content encryption key and the corresponding metadata. To decrypt the content encryption key, the user must be authorized to access the Resource Server using an OAuth 2.0-based mechanism.
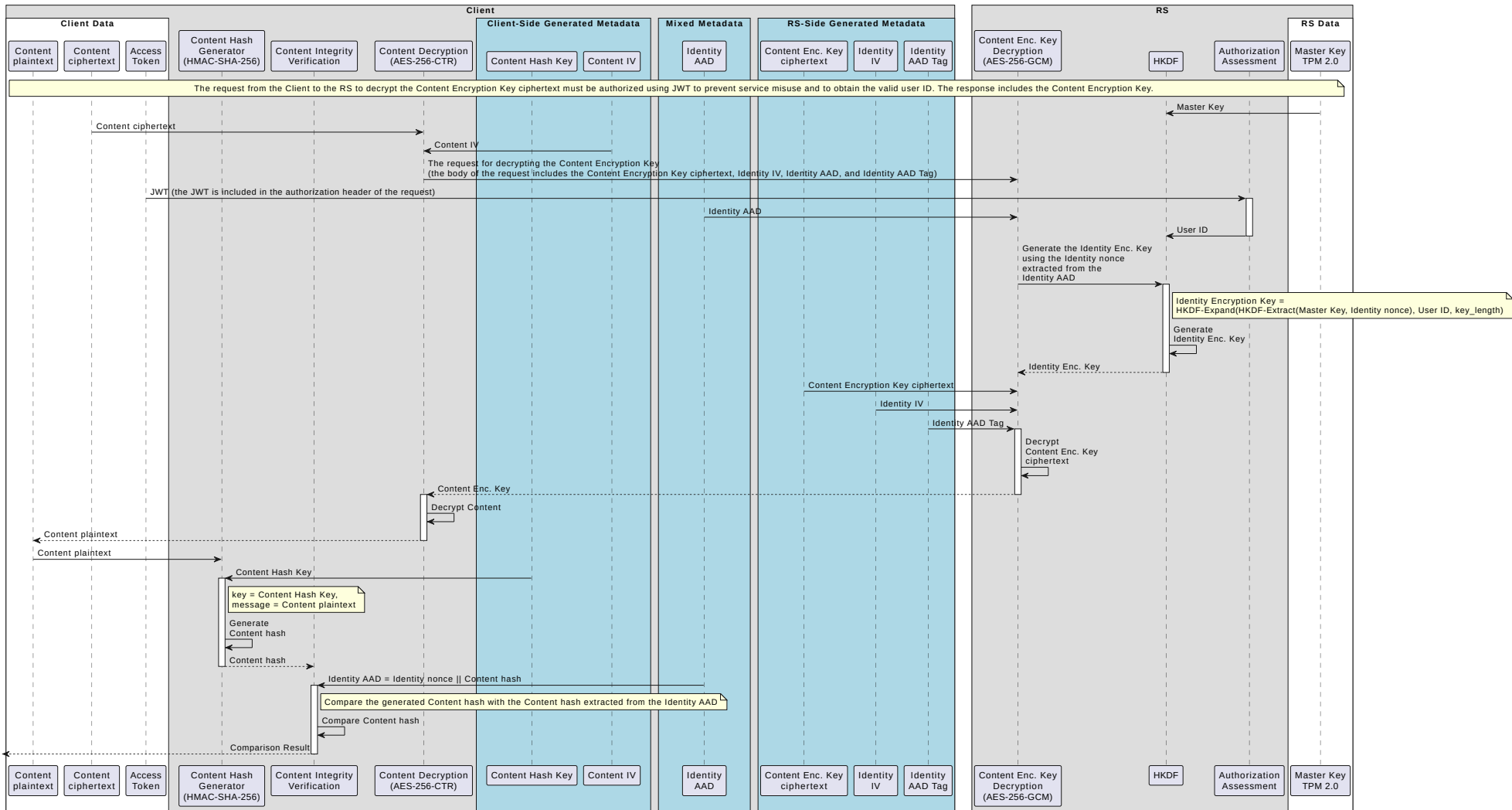
Acronyms

- AS: Authorization Server
- RS: Resource Server

Sequence Diagram for Encryption:

The request from the Client to the RS to encrypt the Content Encryption Key should be authorized using JWT to prevent service misuse. The response includes the Identity nonce, Identity IV, Content Encryption Key ciphertext, and Identity AAD tag.

The AAD may include additional information, typically the URL of the RS.

Master Key

Content plaintext

Get ContentHash Key

Generate Random Content Hash Key

Content Hash Key

Content Hash Key

key = Content Hash Key, message = Content plaintext

Generate Content hash

Content hash

Content plaintext

Get Content Enc. Key

Generate Random Content Enc. Key

Content Enc. Key

User ID (usually the email address of the individual authorized to decrypt the Content Enc. Key and, subsequently, the Content ciphertext)

The request for encrypting the Content Encryption Key (the body of the request includes the Content Encryption Key and the User ID)

Get Content IV

Generate Random Content IV

Content IV

Content IV

Encrypt Content

Content ciphertext

Get Identity Encryption Key

Get Identity nonce

Generate Random Identity nonce

Identity nonce

Identity nonce

User ID

Identity Encryption Key = HKDF-Expand(HKDF-Extract(Master Key, Identity nonce), User ID, key_length

Generate Identity Enc. Key

Identity Enc. Key

Get Identity IV

Generate Random Identity IV

Identity IV

Identity IV

Identity AAD = Identity nonce || Content hash

Encrypt Content Enc. Key

Content Encryption Key ciphertext

Generate AAD Tag

Identity AAD Tag

Sequence Diagram for Decryption:

The decryption process ensures that the decrypted content is indeed the original content that was encrypted for the specific user and verifies that the Resource Server processed the request correctly within the context of the intended user's identity. The authenticity of the Content hash is ensured by the AAD Tag of the AES-256-GCM function and integrity of the Content plaintext is ensured by the HMAC-SHA-256 function.

OAuth 2.0/OIDC PKCE Authorization Code Grant Flow with Audience Restriction and User Email Inclusion
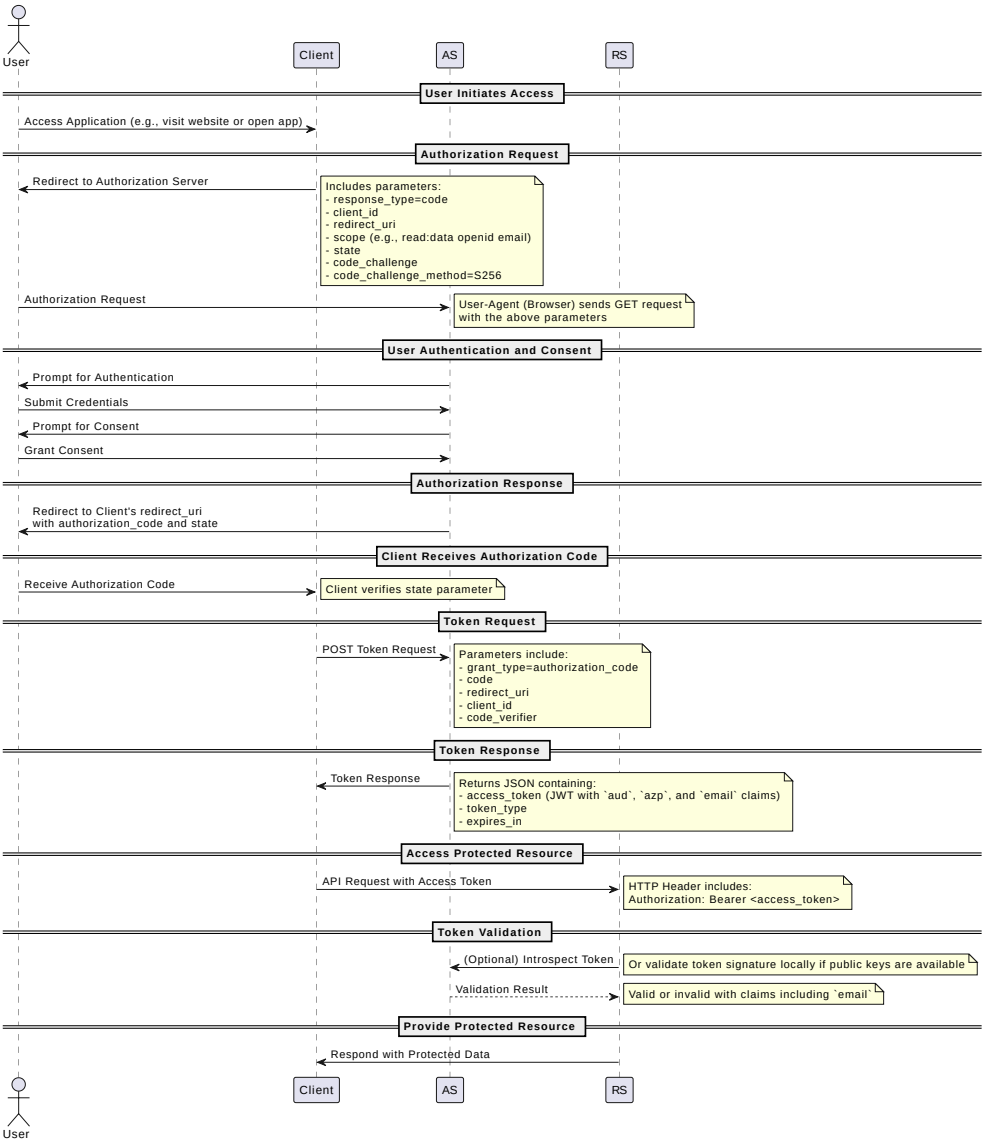
The sequence diagram outlines a mechanism for authenticating users and authorizing access to protected resources using the OAuth 2.0 protocol, enhanced with Proof Key for Code Exchange (PKCE).

Key enhancements in this mechanism include the utilization of the `scope` parameter to define the `aud` (audience) claim within the Access Token, thereby restricting token validity to specific RSs. This ensures that access tokens are purpose-bound, enhancing security by preventing misuse across unintended services. Additionally, the Access Token incorporates the authenticated user's email address as an `email` claim, providing the RS with a reliable identifier for personalized access controls without necessitating additional user information requests. Further strengthening the security framework, the inclusion of the `azp` (Authorized Party) claim within the Access Token binds the token to the specific project or account service ID of the application that requested the token.

The sequence begins with the user initiating access to the client application, which redirects the user to the AS with a detailed authorization request containing parameters such as `scope`, `state`, and PKCE-related values (`code_challenge` and `code_challenge_method`). Upon successful authentication and user consent, the AS issues an Access Token. The Access Token, enriched with the `aud`, `azp`, and `email` claims, is then utilized by the client application to access protected

resources from the RS. It is essential that the RS is properly configured initially to correctly evaluate these claims. The RS validates the token's integrity, audience, and extracts the user's email to manage access appropriately.

Autorization and Identity-Based Symmetric Key Cryptography Mechanisms

In illustrative unrestricted mode, anyone can access the RS and encrypt content for others. However, during the decryption process, authorization is required, and a valid access token with the appropriate claims must be provided. This illustrative mode highlights a specific feature – the decoupling of the encryption initiation from immediate authorization checks at the RS.

Real-World Scenario

We illustrate the process of securely sharing data between Alice and Bob, where Alice wants to share a vacation photo with Bob.

Prerequisites

Alice uses an application (Client) that allows her authenticate via the Authorization Server (AS) and store, retrieve, and share encrypted files on the remote data store (RS1). Additionally, the Client has access to the remote keyring (RS2), which provides identity-based cryptographic functions. The Client has simultaneous access to both the RS1 and RS2. Alice also knows that Bob can obtain authorized access to RS1 and RS2.

Notes:

- The interaction with RS1 is not detailed in this proposal and is considered out of scope.
- In this context, RS2 represents RS in the provided sequence diagrams.
- The Client application acts as the intermediary, interacting with both RS1 and RS2.

Use Case

Alice opens her vacation photo in the Client and enters Bob's email address in the "Share with:" field. She then clicks the "Share" button. The Client encrypts the photo on the client side using RS2's identity-based cryptographic functions and stores the encrypted photo along with its metadata in RS1.

Afterward, Alice notifies Bob that he can access the photo by sending him a shared link to the encrypted file stored in RS1.

Bob can open the shared link after authenticating via AS. His Client retrieves the encrypted photo along with its metadata from RS1, and decrypts it on the client side using RS2's identity-based cryptographic functions. Finally Bob saves the decrypted photo to his local storage.

Notes:

- The encrypted files and associated metadata from the encryption process are persistently stored on RS1 by the Client.
- The keyring can input its URL into the Identity AAD (Identity AAD = Identity nonce || Content hash || Keyring URL) before wrapping the Content Encryption Key. As a result, the Client knows which remote keyring must be used to unwrap the Content Encryption Key ciphertext. This allows Alice to use her dedicated remote keyring, RS2, to encrypt her vacation photo shared with Bob, while Bob can use his remote keyring, RS3, to encrypt his vacation photo shared with Alice, both users using the same remote data store, RS1.
- Alice can have a single dedicated remote Keyring and use multiple sharing services, such as (a) a remote photo store and (b) a remote document repository, each with its own AS provider and respective registered Client. With a properly configured Keyring, and both Clients having Alice's Keyring URL set in their account configuration, such a setup will work without any issues. As long as Bob is registered with both services, Alice can share an encrypted photo and document with him using her single remote Keyring.

Summary

In essence, RS2, RS3 are the crypto engines and key managers, while RS1 is the data and metadata repository. The Client orchestrates the interaction between them.

Conclusion

TBD