Divulgación dinámica y selectiva de identidad basada en pruebas de conocimiento cero

Pablo Santos Cabaleiro
GRADIANT
pscabaleiro@gradiant.org

Xavier Martínez Luaña
GRADIANT
xmartinez@gradiant.org

Jesús Prieto González
GRADIANT
jprieto@gradiant.org

Gonzalo Jiménez Balsa
GRADIANT
gjimenez@gradiant.org

***Resumen*—Las carteras digitales de identidad se encuentran en el foco de muchos investigadores en los últimos años debido a la aparición de paradigmas descentralizados como la identidad autosoberana, la existencia de iniciativas Europeas relevantes como la Infraestructura Europea de Servicios Blockchain (EBSI) o la Cartera Digital Europea de Identidad (EUDIW), y la llegada del reglamento eIDAS 2. Uno de los puntos claves que habilitan estas nuevas soluciones de identidad es la divulgación mínima de información entre carteras, pero por el momento, existen muchos enfoques para conseguirlo, y falta una solución unificada que pueda utilizarse en diversos casos de uso. Debido a esto, en este artículo se propone la integración de un sistema de pruebas de conocimiento cero en los flujos típicos de las carteras digitales alineadas con EBSI y EUDIW, añadiendo la posibilidad de divulgar únicamente la prueba sobre la posesión sobre un atributo de identidad individual de una credencial sin necesidad de revelar el propio atributo ni ninguna otra información contenida en la credencial, mejorando la privacidad de los datos personales del usuario.**

***Index Terms*—Pruebas de Conocimiento Cero, OpenID Connect, Carteras Digitales, Credenciales Verificables**

Tipo de contribución: *Investigación original*

I. Introducción

Los paradigmas de identidad digital autosoberana están ganando relevancia en los últimos años debido a la aparición de iniciativas Europeas como la Infraestructura de Europea de Servicios Blockchain (EBSI) [1] o la Cartera Digital Europea de Identidad (EUDIW) [2], teniendo ambas como objetivo su alineación con la regulación eIDAS2 [3] y GDPR [4]. Uno de los derechos fundamentales que establece la GDPR es la minimización de los datos intercambiados, característica clave que las carteras digitales de identidad pueden ofrecer.

Uno de los enfoques típicos se basa en la gestión de formatos de credenciales que permiten divulgación selectiva, como las credenciales atómicas [5], credenciales basadas en listas de hashes y credenciales con firma multi-mensaje. Estos esquemas revelan únicamente atributos de identidad específicos, pero no permiten demostrar únicamente un hecho sobre ellos, como podrían ser la mayoría de edad o la pertenencia a una provincia. También se ha planteado la creación de atributos específicos que sólo aborden demostraciones de hechos, como es el caso del atributo "ageOverNN" definido en el ARF [6]. El problema de este enfoque es que definir un atributo para cada demostración posible complica enormemente los procesos de definición, implementación y validación de los esquemas de credenciales. Otro problema que acarrean es que las firmas digitales de las credenciales son estáticas, lo que permite trazar a los usuarios, y obliga a emitir varias credenciales con diferentes firmas para poder paliar este problema.

Otra posibilidad son los sistemas y protocolos específicos de divulgación selectiva como VC-FIDO [7], PKIX X.509 [8], Idemix [9] o U-Prove [10], que además de los problemas mencionados anteriormente, son sistemas que divergen de los protocolos de intercambios de credenciales adoptados tanto por EBSI como por EUDIW, que son los flujos de OpenID Connect 4 de emisión [11] y verificación de credenciales [12][13], lo que dificulta su adopción por las carteras digitales.

Finalmente, nos encontramos con las pruebas de conocimiento cero (ZKPs), que a diferencia de los enfoques anteriores, permiten demostrar hechos sobre datos privados, como pueden ser su conocimiento y posesión, que están incluidas en un determinado rango, o que pertenece a un conjunto de valores, sin necesidad de revelar el dato real. Esto convierte a este tipo de técnicas en las más genéricas y con mayor potencial de uso, destacando especialmente las ZK-SNARKs [14] y ZK-STARKs [15]. Las principales limitaciones para aplicar esta tecnología residen en su ineficiencia, tanto en costes de comunicación como de computación, y en la dificultad de integración de estos esquemas dentro de flujos estandarizados de presentación de credenciales. Esto último es clave para su adopción, pues facilita que carteras digitales de diversos proveedores puedan intercambiar ZKPs de manera transparente.

Por todo esto, este artículo propone adaptar de manera eficiente el sistema de identidad basado en CP-SNARKS propuesto en [16], añadiendo una serie de mejoras, como la integración dentro de los flujos de OpenID Connect 4 de intercambio de credenciales [12], la reducción de las dependencias con terceras partes dentro del proceso de presentación de pruebas, y la integración de un mecanismo para evitar que el usuario sea trazado a través de las pruebas sucesivas que ha presentado. Esto implica (i) integrar la fase de configuración del sistema de ZKPs en el flujo de emisión de credenciales verificables, (ii) integrar la fase de generación y verificación de pruebas del sistema de ZKPs en el flujo de presentación de credenciales verificables, (iii) adaptar ambos flujos para reducir su coste de comunicación y su dependencia con terceras partes, y (iv) crear un mecanismo eficiente para evitar vinculación entre diferentes pruebas presentadas por el mismo usuario.

El resto del artículo se organiza de la siguiente manera: en la sección 2 se introduce el trabajo relacionado relevante para nuestra propuesta, incluyendo una pequeña explicación del esquema de ZKPs que se toma como base. Posteriormente, la sección 3 contiene las contribuciones del artículo que se acaban de explicar, y finalmente la sección 4, en la que se

ISBN:978-84-09-62140-8

justifican las decisiones de diseño, se presentan las ventajas y limitaciones identificadas así como las líneas de trabajo futuro planteadas.

II. Trabajo relacionado

Actualmente, existen diferentes mecanismos de divulgación selectiva de atributos [17] para conseguir minimizar la información de las credenciales intercambiada por las carteras digitales. Entre estos, podemos encontrar formatos y protocolos específicos de credenciales con divulgación selectiva así como pruebas de conocimiento cero (ZKPs).

II-A. Formatos de credenciales con divulgación selectiva

Consisten en esquemas de credenciales verificables que permiten presentar atributos individuales de una persona, aunque estos formen parte de una credencial que contenga un conjunto de ellos. Existen varios enfoques para conseguir el efecto descrito.

Las credenciales atómicas [5] contienen un único atributo de información, en lugar de conjuntos de atributos relacionados, lo que permite a una cartera digital presentarlos de una manera selectiva. El problema de este enfoque es que este tipo de credenciales implican la necesidad de definir múltiples esquemas para cada uno de los atributos, y al mismo tiempo, no garantizan que las presentaciones de varias credenciales agrupadas estén emparejadas correctamente. Esto viene motivado principalmente porque puede darse el caso de que atributos con el mismo nombre pueden referirse a diferentes valores y ser usados para engañar a un verificador. Una solución posible a esta problemática es crear atributos universales, pero ello también complica su definición y posterior gestión por parte de las carteras digitales.

Los esquemas de firma con hashes de atributos están basados en tablas indexadas de valores hash, los cuales se guardan junto a las credenciales en la cartera digital del usuario y están directamente relacionados con los atributos de la credencial. Una cartera presenta la credencial con los atributos hasheados junto con los atributos que quiere divulgar, de manera que el verificador puede computar los hashes correspondientes y comprobar que se encuentran dentro de la credencial. Para prevenir la correlación de valores según los hashes, los *salts* de los hashes generados en la fase de emisión deben ser aleatorios y no reutilizables, siendo utilizados una única vez en cada presentación generada. Entre estos esquemas destacan SD-JWT [18], y MSO-mDL [19].

Respecto a las credenciales de firma multi-mensaje, se utilizan algoritmos de firma PS-MS [20], Mercurial [21], CL [22] o BBS [23] que permiten realizar múltiples firmas sobre los atributos de las credenciales verificables, preservando su estructura algebraica durante el intercambio de mensajes, pero cabe destacar que no se considera que tengan seguridad post-cuántica ni se encuentran estandarizadas actualmente.

II-B. Sistemas y protocolos de divulgación selectiva

En la actualidad existen también diferentes sistemas y protocolos que pueden ser utilizados para la divulgación selectiva de atributos. Entre los protocolos utilizados para la divulgación selectiva de atributos mediante credenciales atómicas, podemos destacar VC-FIDO [7] y los certificados de atributos PKIX X.509 [8]; el sistema Idemix, inventado en 2008 por IBM, cuya principal fortaleza es la preservación de la privacidad al autenticarse usando credenciales con atributos [9]; o U-Prove [10], una solución basada en la divulgación selectiva a través del uso de esquemas de firma ciega, la cual define dos protocolos para emisión y presentación de credenciales. Por otro lado, la fundación Hyperledger desarrolló los protocolos de Hyperledger AnonCreds, que están basados en la implementación para credenciales verificables de los protocolos de Hyperledger Indy, Aries y Ursa [25][26][27][28].

II-C. ZKPs

Las Pruebas de Conocimiento Cero (ZKPs) permiten generar demostraciones completas de una afirmación sobre un dato, de manera que un verificador pueda comprobar la validez de dicha afirmación sin revelar ninguna información adicional acerca del dato real utilizado para crear la prueba [29]. Dentro de las pruebas de conocimiento cero distinguimos dos tipos de protocolos, interactivos (IZKP) y no interactivos (NIZKP). La principal diferencia entre ambos reside en la necesidad de crear una conexión simultánea entre el poseedor de la credencial y el verificador. Los protocolos no interactivos presentan ventajas de flexibilidad y eficiencia en términos de tiempo y recursos computacionales, lo que mejora experiencia de usuario. Dentro de estos, podemos destacar tres conjuntos de protocolos que cumplen con las medidas de seguridad necesarias para garantizar una divulgación selectiva de atributos segura: BulletProofs [30], ZK-STARK [15] y ZK-SNARK [14]. Las BulletProofs se basan en un protocolo ZKP que busca reducir el tamaño de las transacciones y tiempos de verificación en los protocolos existentes. Sus principales limitaciones residen en que únicamente permiten generar pruebas sobre rangos de valores y que no se considera que tengan seguridad post-cuántica. Por otro lado, las ZK-SNARKs se basan en protocolos diseñados para proveer pruebas de predicados, divulgación selectiva y protección frente a la trazabilidad de los datos del usuario, siendo su principal limitación la capacidad computacional necesaria y las restricciones de tamaño necesarias, siendo éstas dependientes de la demostración que se desee generar. Finalmente, las ZK-STARK, evolución de los protocolos anteriores, permiten crear pruebas de menor tamaño y por lo tanto más sencillas de verificar. Además, éstas son consideradas más transparentes, pues cualquier entidad puede verificar la prueba de conocimiento cero sin necesidad de solicitar información adicional. Su principal limitación reside en que se trata de un campo poco explorado, y con pocos algoritmos propuestos en la actualidad. La seguridad post-cuántica de las ZK-SNARKS y las ZK-STARKS depende completamente de las funciones Hash y algoritmos de firma utilizados.

Como solución a las limitaciones de eficiencia y escalabilidad de las ZK-SNARKs anteriormente comentadas, surge el protocolo Commit and Prove SNARKs (CP-SNARKs), el cual propone un nuevo procedimiento en el proceso de generación y validación de las pruebas de conocimiento cero, denominados *commitments*. El proceso de generación y validación de estos *commitments* puede ser un inconveniente en términos de complejidad computacional, tamaño de éstos, y por ende, tiempo de validación, ya que son construidos utilizando ecuaciones matemáticas que involucran valores secretos y públicos del usuario en este caso. Para solventar esta limitación, este

protocolo propone generar un valor adicional, el cual es parte de la prueba de conocimiento cero, denominado *CP-SNARK*. Este valor permite validar que un *commitment* está correctamente construido y que además procede del conocimiento del atributo, pero sin que se revele su valor real. De esta manera, en lugar de verificar la corrección del *commitment* dentro de la relación de prueba, se verifica fuera de ella, lo que reduce tanto la carga computacional como el tamaño de la misma [16].

III. Diseño de la solución

En esta sección se presentan el diseño, organización y funcionamiento de la solución propuesta. Comienza introduciendo los componentes principales, analizando su rol principal, y cómo interactúan entre ellos en los diferentes flujos operativos de emisión, autenticación y verificación. Los flujos definidos se encuentran alineados con los estándares de OpenId Connect de emisión de credenciales verificables (OIDC4VCI) [11] y de presentación de credenciales verificables (OIDC4VP) [12], así como con las especificaciones de conformidad con la Cartera Digital Europea de Identidad (EUDIW) [2] y la Infraestructura de Europea de Servicios Blockchain (EBSI) [1]. Durante la explicación de los flujos operativos se detalla la integración del protocolo de ZKPs al flujo definido, la adaptación de los diferentes parámetros y su funcionamiento, desde la solicitud hasta la verificación de la credencial de conocimiento cero.

III-A. Arquitectura

La arquitectura general de la solución propuesta se compone principalmente de seis actores:

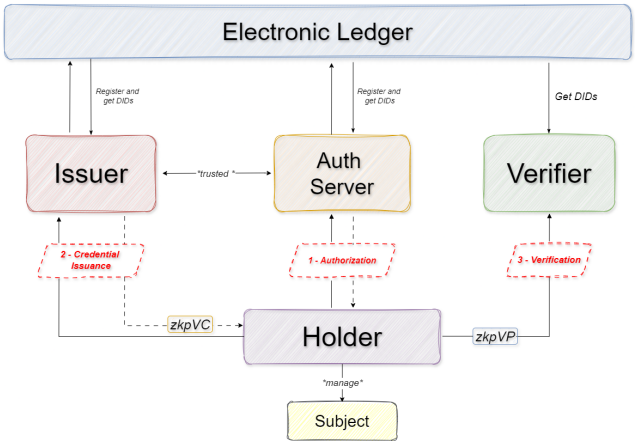

Figura 1. Diagrama general de arquitectura

1. **Electronic Ledger:** Registro digital seguro e inmutable utilizado para almacenar información de manera transparente y descentralizada sobre el resto de componentes, en este caso Identificadores Descentralizados (DIDs) [31], con propósitos de verificación de información.
2. **Issuer:** Institución o autoridad con capacidad de emisión de credenciales verificables válidas y confiables.
3. **Authorization Server:** Entidad confiable encargada de realizar el proceso de autenticación y autorización del usuario. Es frecuente que las figuras de *Issuer* y *Authorization Server* sean representadas por la misma entidad.
4. **Verifier:** Entidad o individuo encargado de verificar la autenticidad e integridad de las credenciales digitales presentadas por un *holder*.
5. **Subject:** Entidad, individuo u objeto sobre el cual se emite una credencial verificable.
6. **Holder**: Entidad o individuo el cual posee o administra credenciales y responsable de la solicitud, gestión y presentación de credenciales verificables.

Por último cabe mencionar dos estructuras de datos que presentaremos posteriormente, como son las Credenciales Verificables de Conocimiento Zero (*zkpVC*) o las Presentaciones Verificables de Conocimiento Cero (*zkpVP*).

III-B. Flujos Operativos

Como hemos introducido anteriormente en la figura 1, el flujo principal de la solución, basado en los estándares de emisión [11] y presentación [12] de credenciales, se compone de 3 flujos principales: *Authorization*, donde el usuario realiza un proceso de autenticación y autorización contra el *Auth Server*; *Credential Issuance*, en el cual se produce la solicitud y la emisión de la credencial de conocimiento cero entre el *holder* y el *issuer*; y *Verification*, en donde el *verifier* lleva a cabo el proceso de validación de la presentación verificable de conocimiento cero creada por el *holder*.

Adicionalmente, existe un proceso opcional previo al proceso de autorización, denominado "Ofrecimiento y Descubrimiento", en el que el *issuer*, mediante dos endpoints expuestos, propone el inicio del flujo e intercambia la información pertinente con el *holder* para la emisión de credenciales garantizando un proceso cómodo y seguro [11].

III-B1. Credential Offering e Issuer Metadata Discovery: En esta sección exploraremos la fase opcional de inicialización del flujo de emisión de credenciales. Esta fase consta principalmente de dos procesos fundamentales: el *Credential Offering* y el *Issuer Metadata Discovery*. A continuación se muestra un esquema en el que se indican los diferentes parámetros intercambiados durante las interacciones de ambos procesos:

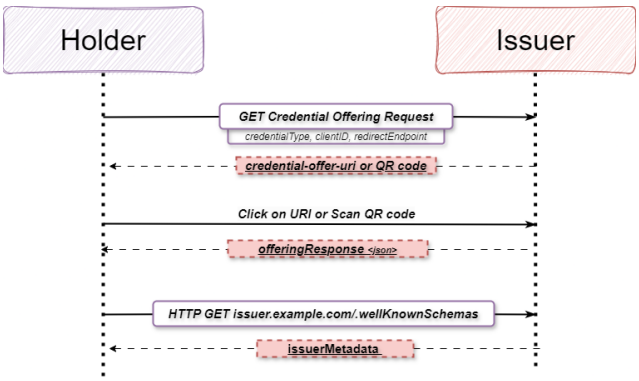

Figura 2. Diagrama de flujo del flujo de inicialización

El flujo comienza con el ofrecimiento de credenciales o "Credential Offering", proceso en el que el *holder* define

los parámetros iniciales del flujo de emisión de credenciales, indicando al *issuer* el tipo de credencial que desea solicitar de entre las disponibles, y el DID [31] al que desea que la credencial sea asociada. Con estos valores, el *issuer* ofrecerá, mediante el endpoint correspondiente, la posibilidad de iniciar el flujo de emisión de las credenciales.

En esta oferta, el *issuer* proporciona como respuesta un objeto JSON en el que se indican los parámetros escogidos por el *holder* e iniciando el flujo de autorización. Cabe destacar la posibilidad de proporcionar dicho objeto directamente, bien a través de una URL o bien un código QR que deberá ser escaneado por el *holder* a través de su *wallet*.

A continuación, se indica un ejemplo de una petición de inicio del flujo de emisión de una credencial de conocimiento cero y su correspondiente respuesta "offeringResponse" con la información proporcionada por el *issuer*:

HTTP GET

https://issuer.example.com/initiate-credential-offer
?**credential_type**=ZkpVerifiableCredential
&**client_id**=holderDID

```
{
    "credential_issuer": "issuer.example.com",
    "credentials": [
        {
            "format": "jwt_vc",
            "types": [
                "VerifiableCredential",
                "ZkpVerifiableCredential"
            ],
        }
    ],
    "issuer_state":"eyjfaoiuf631h..."
}
```

Como puede observarse en el ejemplo expuesto, se define un nuevo tipo de credencial "ZkpVerifiableCredential" solicitada por el usuario. Al final del proceso de emisión, esta credencial contendrá el *commitment* correspondiente a un atributo de identidad junto con la firma del *issuer* verificando que la información definida en el *commitment* es verídica.

Una vez finalizado el proceso de ofrecimiento de credenciales, el *holder* continúa con el proceso de "*Issuer Metadata Discovery*" o "Descubrimiento de Metadatos del *Issuer*".

Este proceso permite al *holder* obtener información sobre las diferentes características y capacidades del *issuer* en la emisión de credenciales, entre las que destacan los endpoints definidos para cada proceso de la emisión de credenciales o los diferentes algoritmos de cifrado, tipos y formatos de credenciales soportados.

Para ello, al igual que en el proceso anterior, el *holder* debe realizar una petición al endpoint expuesto por el *issuer* para proporcionar dicha información. De acuerdo con el estándar [11], el *holder* debe obtener la dirección del endpoint añadiendo el string "*/.well-known/openid-credential-issuer*" al valor recibido en el campo "*credential_issuer*" del "*Credential Offering*".

A continuación se muestra un ejemplo del proceso de descubrimiento de metadatos del *issuer* mediante la interacción del *holder* con el endpoint correspondiente:

HTTP GET

https://issuer.example.com/.well-known/openid-credential-issuer

```
"credential_issuer": "https://issuer.com",
"authorization_servers": ["https://authserver.
    com/authorize"],
"credential_endpoint": "https://issuer.com/
    credential",
"batch_credential_endpoint": "https://issuer.com
    /batch_credential",
"deferred_credential_endpoint": "https://issuer.
    com/deferred_credential",
"credential_configurations_supported": {
    "ZeroKnowledgeCredential": {
        "format": "jwt_vc",
        "scope": "openid",
        "credential_definition":{
            "type": [
                "VerifiableCredential",
                "VerifiableAttestation",
                "ZKPVerifiableCredential"
            ]
}}}
```

Como podemos observar, en la respuesta, se incluyen los diferentes endpoints con los que el *holder* interactúa en los diferentes procesos del flujo general, así como los diferentes tipos de credenciales soportadas por el *issuer*. Entre los endpoints definidos, podemos destacar el endpoint destinado para realizar el flujo de autorización "*authorization_servers*", el endpoint en el que se define el flujo de emisión de credenciales "*credential_endpoint*" o el endpoint definido para el caso en el que el usuario desee recibir un lote de credenciales "*batch_credential_endpoint*". De entre las credenciales soportadas, se incluye el nuevo tipo definido "*ZKPVerifiableCredential*".

III-B2. Authorization: El flujo de autorización se define con el objetivo de realizar un proceso de autenticación del usuario contra la entidad organización designada en cada caso. Durante este proceso, el *holder* intercambia una serie de peticiones y tokens con el *authorization_server*, de manera que, al finalizar este proceso, el usuario recibirá un token de acceso firmado por la entidad autorizadora que le permitirá solicitar la credencial al *issuer* en el endpoint definido para ello. En este caso, como hemos introducido anteriormente, el proceso de autorización establecido para la solución está totalmente alineado tanto con el estándar OIDC4 [11] y con EBSI [1], lo cual garantiza su integrabilidad con el resto de sistemas alineados con estos estándares.

El proceso de autorización definido comienza con la solicitud de la autorización por parte del *holder* al endpoint de autorización definido por el *authorization_server*. Como respuesta, el servidor le proporciona un *"Token Request"* en el que se incluye la información de la petición realizada, la cual el usuario debe utilizar para o bien construir el token solicitado, o bien obtenerlo a través de un servicio de identificación válido. Típicamente, el servidor proporcionará una *"VP Token Request"*, en la que solicitará el valor original a través de una presentación de la credencial que contiene dicho atributo, aunque cualquier otro método de identificación

podría ser utilizado. Para el caso de las presentaciones, se recomienda utilizar SD-JWT [18] para sólo compartir el atributo requerido y proteger la privacidad del resto de datos privados contenidos en la credencial. Tras acabar este proceso, el *authorization_server* devolverá un código de autorización (*Authorization_Code*) que el holder enviará al endpoint correspondiente junto con otros parámetros de seguridad necesarios [11] para obtener el token de acceso firmado por la entidad autorizadora.

A continuación, se muestra un ejemplo de una solicitud y respuesta de autorización:

HTTP GET /authorize

?**response_type**=code
&**scope**=ZKPVerifiableCredential
&**resource**=https://credential-issuer.example.com
&**client_id**=did:key:JguCHao...
&**code_challenge**=E9Melhoa2OwvFrEMTu
&**code_challenge_method**=S256
&**redirect_uri**=http://client.example.org/

De la petición anterior podemos destacar el parámetro "*scope*", en el cual el usuario define el tipo de credencial que desea solicitar. En caso de que el usuario haya seguido el flujo de Ofrecimiento/Descubrimiento (III-B1) deberá indicar dicho valor mediante el parámetro *authorization_details*, añadiendo aquellos valores recibidos en dicho proceso. A continuación se muestra un ejemplo de esta casuística:

HTTP GET /authorize

response_type=code
&**client_id**=s6BhdRkqt3
&**code_challenge**=E9Me2OwvFt8UR...
&**code_challenge_method**=S256
&**authorization_details**= [”type”: “openid_credential”, “credential_configuration_id”: “ZKPVerifiableCredential”]
&**redirect_uri**=https://client.example.org/

Tras realizar esta petición, en caso de que el proceso haya sido satisfactorio, el *authorization_server* le facilita al *holder* el código de autorización.

HTTP/1.1 302 Found Location

Location: https://wallet.example.org/cb
?**code**=SplxlOBeZQQYbYS6WxSbIA

Con este código, el usuario puede demostrar haber superado con éxito el proceso de autorización y posteriormente podrá intercambiarlo por un token de acceso o *AccessToken*. Dicho token se encuentra firmado por la entidad autorizadora y, debido a que entre esta entidad y el *issuer* existe una relación de confianza establecida a través del modelo de gobernanza ubicado en el *electronic ledger*, con él podrá demostrar al *issuer* que ha completado exitosamente el proceso de autenticación presentando dicho token. Al mismo tiempo, el *issuer*, podrá comprobar la validez del token verificando la firma del token y consultando la información pública del *authorization server* almacenada en el *electronic ledger*.

Cabe recordar, que, como se ha mencionado anteriormente, el *issuer* y el *Authorization Server* pueden ser la misma entidad. Adicionalmente, el estándar define un flujo alternativo, denominado "*Pre-Authorized Code Flow*" [11], en el cuál el *holder* ya se encuentra autenticado ante el emisor, y puede saltarse esa fase del flujo, pasando directamente a la solicitud de la credencial.

III-B3. Credential Issuance: Una vez el usuario se ha autenticado y ha sido autorizado, o bien ya lo estaba (flujo *Pre-Authorized*), podrá solicitar al *issuer* la emisión de una credencial de conocimiento cero. A continuación en la Figura 3 se muestra un diagrama de los diferentes procesos que componen este flujo:

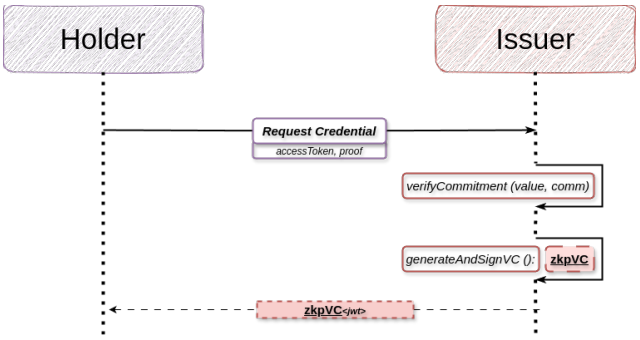

Figura 3. Diagrama de flujo de expedición de credenciales

El proceso comienza con el *holder*, quien a través de una petición al endpoint correspondiente definido por el *issuer*, inicia el proceso de emisión de credenciales. Junto con dicha petición, el usuario debe proporcionar dos valores:

1. *Access Token:* Token de acceso firmado emitido por la entidad encargada del proceso de autenticación correspondiente el cual permite iniciar el proceso de emisión de credenciales. Este token contendrá el valor real del atributo sobre el que usuario quiere generar pruebas.
2. *Proof*: Objeto que contiene la prueba de posesión de la clave criptográfica a la que estaría vinculada la Credencial expedida. Debe incorporar el Identificador del Emisor de Credenciales (audiencia), un valor "c_nonce" generado por el Servidor de Autorización para permitir al Emisor de Credenciales detectar la repetición [11], y un *commitment (comm)*, el valor criptográfico que permite demostrar la posesión de un atributo sin necesidad de revelar su valor.

Es importante mencionar que el *issuer* debe verificar que el *commitment* que aparece en el campo *proof* se corresponde con el dato real del usuario, para posteriormente introducirlo en la credencial verificable de conocimiento cero (*zkpVC*). Este dato real normalmente llegará dentro del *Access Token*

introducido por el *Authorization Server*, aunque se podrían plantear flujos alternativos en los que el este actor no obtuviese dicho valor, y el usuario se lo hiciera llegar al emisor de otra forma confiable (p. ej. presentación de una credencial).

A continuación se muestra un ejemplo de solicitud de credencial en la que el *holder* envía los valores anteriormente mencionados al *issuer*.

HTTP POST /credential

Host: issuer.example.com
Content-Type: application/json
Authorization: BEARER <**accessToken**>

{
“format”: “jwt_vc”,
“types”: [“string”],
“proof”: { “proof_type”: “jwt”, “jwt”: “eyJ0eX...”},
}

Como respuesta, el usuario obtendrá la credencial de conocimiento de cero firmada (*zkpVC*) la cual contendrá la sentencia demostrada junto con el *commitment* correspondiente.

Pongamos el caso de que el *holder* desea obtener una credencial verificable de conocimiento cero, con la que pueda demostrar que es mayor de edad sin necesidad de revelar el resto de sus atributos de identidad al verificador en cuestión. Partiendo de la base de que el *holder* tiene en posesión una credencial verificable con la que puede demostrar su fecha de nacimiento crea una presentación verificable. Además, genera el commitment de su fecha de nacimiento, y realiza el proceso de autorización y emisión de credenciales descritos para obtener la credencial verificable de conocimiento cero (*zkpVC*). A continuación, se muestra un ejemplo de *zkpVC* decodificada para el caso de uso propuesto:

```
{
  "alg": "ES256",
  "typ": "JWT"
}
{
  "iat": "1357817395824",
  "type": [ZKPVerifiableCredential]
  "format":"jwt_vc"
     ...
  "credentialSubject": {
      "commitment": [
          {
              "attribute": "dateOfBirth",
              "value": "<commitment>"
          }
      ]
  }
}
```

Debido a que el usuario puede presentar su credencial de conocimiento cero ante diferentes servicios (o varias veces dentro del mismo servicio), éstos tienen la capacidad de identificar al usuario mediante la propia credencial. Para evitar esto, el usuario puede solicitar la emisión de más de una credencial de conocimiento cero utilizando el endpoint definido para ello (ver *Batch Credential Endpoint [11]*), introduciendo varios *commitments* generados sobre el atributo personal para que el *issuer* emita una credencial para cada uno de ellos. De esta manera, podrá presentar diferentes credenciales para el mismo propósito, evitando la trazabilidad de su actividad por parte de los diferentes servicios con los que interactúa.

III-B4. Verification: Una vez el usuario posee su credencial de conocimiento cero tiene la capacidad de demostrar la información contenida en ella ante una entidad verificadora, con la característica de que podrá demostrar la posesión de un atributo o la veracidad de una sentencia asociada a su identidad sin necesidad de revelar información personal sensible y privada. En el diagrama de flujo presentado a continuación es posible observar las diferentes interacciones y procesos realizados de cada una de los actores:

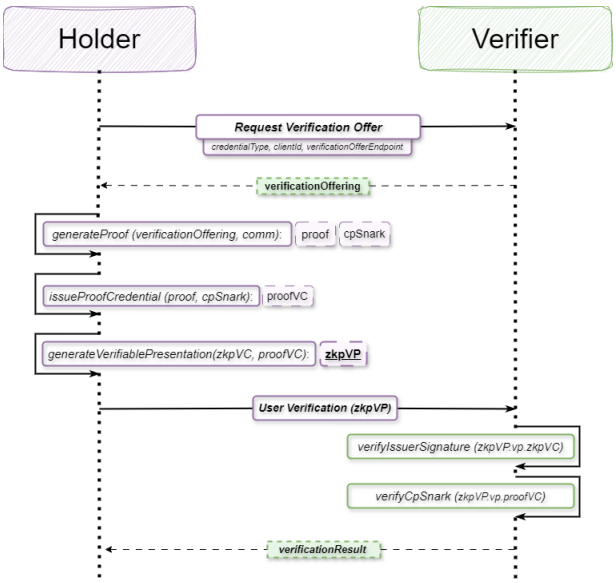

Figura 4. Diagrama de flujo de verificación

El flujo comienza con el proceso de ofrecimiento de verificación, prácticamente idéntico al explicado anteriormente III-B1, en el que el *verifier* dispone de un endpoint dedicado para iniciar el flujo de verificación con un usuario final. De esta manera, el usuario deberá indicar el tipo de credencial que desea validar, así como el DID [31] asociado a dicha credencial, obteniendo como respuesta un objeto “*verificationOffering*”. A continuación, se muestra un ejemplo de “*verificationOffering*” obtenido para el caso de uso en el que el *verifier* ha solicitado validar la mayoría de edad del usuario.

```
{
  "alg": "ES256",
  "typ": "JWT"
  "kid": "...."
}

{
   "credential_issuer": "...",
   "credentials": [
       {
           "format": "jwt_vc",
           "types": [
               "VerifiableCredential",
               "VerifiableAttestation",
               "ClaimProofCredential",
           ],
       "verificationClaim": [ <VerificationOBJ> ]
```

```
        },
        {
            "format": "jwt_vc",
            "types": [
                "VerifiableCredential",
                "VerifiableAttestation",
                "ZKPVerifiableCredential"
            ]
        }
    ],
    "grants": {
        "authorization_code": {
            "issuer_state": "..."
        }
    }
}
```

En esta respuesta, el *verifier* indica los tipos de credenciales que el *holder* debe presentar para realizar la verificación, así como la función de validación que se deberá aplicar a los datos a que se desean verificar. Para el caso de uso propuesto, este objeto seguiría una estructura similar a la siguiente:

```
"verificationClaim": [
    {
        "type": "moreThan"
        "value": "18"
    },
]
```

Con esta información el *holder* deberá generar una prueba de conocimiento cero, que permita demostrar que cumple el claim requerido sin necesidad de revelar el dato en cuestión. Para ello tras recibir la función de validación requerida en el "verificationClaim" (en este caso, la sentencia de que es mayor de edad), transformarla en una expresión matemática y aplicar el *commitment* generado anteriormente a la función, obteniendo el resultado de la función (*proof*) y la *cpSnark* asociada a dicha prueba. Este último valor permite a un tercero verificar que dicha prueba está bien construida, así como que el *commitment* correspondiente es generado a partir del conocimiento del atributo sin revelar su valor real [16].

Una vez generada la prueba de conocimiento cero, junto con su correspondiente *cpSnark*, el *holder* emite una credencial verificable autofirmada ("*proofVC*"), en la que incluirá el *proof* y el *cpSnark* comentados anteriormente, posteriormente compartirlo con el *verifier* en el proceso de verificación.

Con este propósito, el *holder* creará una presentación verificable de conocimiento cero (*zkpVP*) en la que añadirá tanto la credencial que incluye la prueba de conocimiento cero generada (*proofVC*) como la credencial verificable emitida por el *issuer* (*zkpVC*). A continuación se muestra un ejemplo de *zkpVP* para el caso de uso propuesto:

```
{
"iat": "1357817395824"
"exp": 1357817395824 ,
"response_type": "vp_token"
 "vp": {
    '@context': ['https://www.w3.org/2018/
        credentials/v1']
        id: 'urn:uuid:0706061a...',
        type: ['VerifiablePresentation'],
        holder: 'did:key:z2dmzD81cgPx8Vki...
        verifiableCredential: [
            {
                "type": "ZKPCredential"
                "value": <zkpVc>
            },
            {
                "type": "ClaimProofCredential"
                "value": <proofVC>
            }
        ]
    }
}
```

Una vez recibida, el *verifier* comenzará con el proceso de validación de los datos incluidos en la presentación verificable de conocimiento cero (*zkpVP*). Primeramente verificando la firma del *issuer* y del *holder* de cada una de las credenciales incluidas, y posteriormente validando la prueba de conocimiento cero a través de la *cpSnark*. En función de los resultados obtenidos en las validaciones, el *verifier* proporciona un objeto "*verificationResult*" a través del cual le indica al usuario el resultado del proceso de verificación.

En este punto, el *holder* ha completado el proceso tanto de emisión como verificación de atributos utilizando pruebas de conocimiento cero, que le han permitido demostrar la posesión de un atributo o la afirmación de una sentencia sin necesidad de revelar a la entidad verificadora ningún atributo privado de su identidad digital.

IV. Conclusiones

En este artículo, hemos propuesto un sistema de divulgación selectiva de información basado en ZKPs capaz de demostrar hechos sobre la información privada del usuario de manera segura y eficiente. El sistema es completamente integrable con los flujos estandarizados de emisión y verificación de credenciales de OpenID Connect 4 [11], y permite generar pruebas sucesivas no vinculables entre si, evitando que la actividad del usuario sea trazable por los servicios con los que interactúa y añadiendo un mayor nivel de privacidad con respecto a otras soluciones propuestas. Adicionalmente el sistema propuesto, eliminando las dependencias existentes entre las terceras partes, permite al usuario disponer de una mayor autonomía en los procesos de emisión y verificación de su identidad digital.

Como hemos introducido anteriormente, la arquitectura, componentes y funcionamiento de la solución siguen los patrones establecidos por el estándar de emisión de credenciales verificables [11], presentaciones verificables [12] y EBSI [1]. La alineación con estos estándares permite a la solución propuesta ser totalmente integrable y adaptable a todas aquellas soluciones que tomen como referencia las directrices definidas en estos documentos.

Por otro lado, la selección de las zk-SNARKS como protocolo de pruebas de conocimiento cero permite grandes ventajas en términos demostración dinámica de hechos sobre credenciales, e integrabilidad con el estándar de emisión de credenciales verificables, pues las características intrínsecas de las zk-SNARKS permiten que no sea necesaria la interacción entre las entidades emisoras y verificadoras.

Por último, cabe destacar la elección de las cp-SNARKS como mecanismo de verificación de las pruebas de conocimiento cero. La implementación de esta solución permite obtener una mejor eficiencia computacional y de comunicación que otras alternativas dentro del mismo campo, ya que separa la verificación de la firma digital del circuito aritmético a demostrar, obteniendo una reducción considerable del tamaño de la prueba, y la consiguiente reducción de coste computacional y de comunicación.

La limitación principal identificada durante el desarrollo del trabajo es la complejidad adicional que añade esta propuesta a la información que gestionan las carteras digitales. Otro aspecto a evaluar como una limitación es la seguridad post-cuántica del sistema, que en caso de ser vulnerable, habría una dificultad añadida para proteger el esquema sin afectar a sus propiedades. Este análisis se encuentra fuera del alcance del presente documento.

Las posibles líneas de trabajo futuro comprenden la implementación del sistema propuesto en carteras digitales en un entorno real, así como su aplicación a un caso de uso, como podría ser demostrar la mayoría de edad, nacionalidad o cualquier otro atributo de un usuario con diferentes fines, como el cumplimiento de regulaciones, la participación en sistemas de votación o el acceso a información restringida, de una manera más privada. Adicionalmente, también se podría explorar el uso de Entornos de Ejecución Confiables (TEE) [32] por parte de los actores del sistema propuesto con el objetivo de aumentar la privacidad del usuario durante todo el proceso descrito.

Agradecimientos

Este trabajo es parte del proyecto “creación y gestión de Identidades digitales Basadas En Registros distribuidos e Inteligencia Artificial (IBERIA)” , perteneciente la Iniciativa Estratégica de Compra Pública de Innovación, CPP001/22, del Instituto Nacional de Ciberseguridad (INCIBE) [33].

Referencias

[1] European Commission (EC) y European Blockchain Partnership (EBP): “How to issue Verifiable Credentials”, en *https://hub.ebsi.eu/conformance/learn/verifiable-credential-issuance*, 21 Febrero 2024.

[2] EUDI Wallet Consortium: “EU Digital Identity Wallet Consortium”, en *https://eudiwalletconsortium.org/*, 4 Febrero 2024.

[3] Parlamento Europeo y el Consejo de la Union Europea: “Propuesta por la que se modifica el Reglamento (UE) n.º 910/2014 en lo que respecta al establecimiento de un Marco para una Identidad Digital Europea”, en *https://eur-lex.europa.eu/legal-content/ES/TXT/?uri=CELEX%3A52021PC0281*, 6 Junio 2021.

[4] Parlamento Europeo y el Consejo de la Union Europea: “REGLAMENTO (UE) 2016/ 679 ”, en *https://eur-lex.europa.eu/legal-content/ES/TXT/PDF/?uri=CELEX:32016R0679*, 27 Abril 2016.

[5] Manu Sporny, Dave Longley y David Chadwick: “Verifiable Credentials Data Model v1.1”, en *https://www.w3.org/TR/vc-data-model*, 2 Marzo 2022.

[6] EU Digital Identity Wallet: “EUDI Document Architecture and Reference Framework”, en *https://github.com/eu-digital-identity-wallet/eudi-doc-architecture-and-reference-framework/blob/main/docs/arf.md*, 10 Febrero 2023.

[7] David W. Chadwick: “The Use of FIDO2 and Verifiable Credentials”, en *https://www.youtube.com/watch?v=l3taGxBdrRU*, 12 Septiembre 2020.

[8] Sean Turner, Stephen Farrell y Russ Housley: “An Internet Attribute Certificate Profile for Authorization”, en *IETF RFC 5755*, Enero 2010.

[9] IMB Research: “Identity Mixer (IDEMIX)”, en *IBM Zurich*, Noviembre 2015.

[10] Stefan Brands: “Rethinking Public Key Infrastructures and Digital Certificates; Building in Privacy”, en *MIT Press*, 30 Agosto 2000.

[11] T.Lodderstedt, K.Yasuda, T.Looker: “OpenID for Verifiable Credential Issuance - draft 13”, en *https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html*, 8 Febrero 2024.

[12] O.Terbu , T.Lodderstedt, K.Yasuda, T.Looker : “OpenID for Verifiable Presentations - draft 20”, en *https://openid.net/specs/openid-4-verifiable-presentations-1_0.html*, 29 Noviembre 2023.

[13] K. Yasuda, M. Jones, T. Lodderstedt: “Self-Issued OpenID Provider v2 - draft 13”, en *https://openid.net/specs/openid-connect-self-issued-v2-1_0.html*, 28 Noviembre 2023.

[14] Nir Bitansky, Ran Canetti, Alessandro Chiesa y Eran Tromer: “From Extractable Collision Resistance to Succinct Non-Interactive Arguments of Knowledge, and Back Again”, en *Information Technology Convergence and Services*, 30 Noviembre 2011.

[15] Maksym Petkus: “Why and How zk-SNARK Works: Definitive Explanation”, en *arXiv:1906.07221*, 17 Junio 2019.

[16] Jeonghyuk Lee, Jaekyung Choi, Jihye Kim y Hyunok Oh: “Privacy-preserving Identity Management System”, en *Cryptology ePrint Archive: https://eprint.iacr.org/2021/1459*, 6 Noviembre 2021.

[17] Electronic Signatures and Infrastructures (ESI): “Analysis of selective disclosure and zero-knowledge proofs applied to Electronic Attestation of Attributes”, en *DTR/ESI-0019476*, August 2023.

[18] Daniel Fett, Kristina Yasuda y Brian Campbell: “Selective Disclosure for JWTs (SD-JWT)”, en *https://datatracker.ietf.org/doc/html/draft-ietf-oauth-selective-disclosure-jwt-07*, 11 Diciembre 2023.

[19] ISO/IEC JTC 1/SC 17: “Personal identification - ISO-compliant driving licence - Part 5:Mobile driving licence (mDL) application”, en *ISO/IEC 18013-5*, Septiembre 2021.

[20] David Pointcheval y Olivier Sanders: “Short Randomizable Signatures”, en *RSA Conference*, 4 marzo 2016.

[21] Elizabeth C. Crites y Anna Lysyanskaya: “Mercurial Signatures for Variable-Length Messages”, en *PET Symposium*, 18 Agosto 2020.

[22] Jan Camenisch y Anna Lysyanskaya: “A Signature Scheme with Efficient Protocols”, en *Security in Communication Networks, Third International Conference, SCN*, 11 Septiembre 2002.

[23] Dan Boneh, Xavier Boyen y Hovav Shacham: “Short Group Signatures”, en *Annual International Cryptology Conference*, 15 Agosto 2004.

[24] Jeremie Miller, David Waite y Michael B. Jones: “JSON Web Proof (JWP)”, en *https://datatracker.ietf.org/doc/html/draft-ietfjose-json-web-proof-02*, 21 Octubre 2023.

[25] Stephen Curran, Artur Philipp, Hakan Yildiz, Sam Curren, Victor Martinez Jurado, Aritra Bhaduri y Artem Ivanov: “AnonCreds Specification v1.0”, en *https://github.com/hyperledger/anoncreds-spec*

[26] Hyperledger Foundation: “Hyperledger Indy”, en *https://www.hyperledger.org/projects/hyperledger-indy*

[27] Hyperledger Foundation: “Hyperledger Aries”, en *https://www.hyperledger.org/projects/aries*

[28] Hyperledger Foundation: “Hyperledger Ursa”, en *https://wiki.hyperledger.org/display/ursa*

[29] Mohameden Dieye, Pierre Valiorgue, Jean-Patrick Gelas, El-Hacen Diallo, Parisa Ghodous, Frédérique Biennier y Éric Peyrol: “A self-Sovereing Identity Based on Zero-Knowledge Proof and Blockchain”, en *10.1109/ACCESS.2023.3268768*, vol. 11. 20 Abril 2023.

[30] Benedikt Bünz, Jonathan Bootle y Dan Boneh: “Bulletproofs: Short Proofs for Confidential Transactions and More”, en *2018 IEEE Symposium on Security and Privacy (SP)*, 20 Mayo 2018.

[31] Manu Sporny, Dave Longley, Markus Sabadello ,Drummond Reed ,Orie Steele, Christopher Allen: “Decentralized Identifiers (DIDs) v1.0”, en *OpenId Connect: https://www.w3.org/TR/did-core/*, 19 Julio 2022.

[32] Mohamed Sabt; Mohammed Achemlal; Abdelmadjid Bouabdallah: “Trusted Execution Environment: What It is, and What It is Not”, en *https://ieeexplore.ieee.org/document/7345265*, 03 Agosto 2015.

[33] Instituto Nacional de Ciberseguridad: “Compra Pública de Innovación” , en *https://www.incibe.es/industria-cpi*, 9 Mayo 2024.