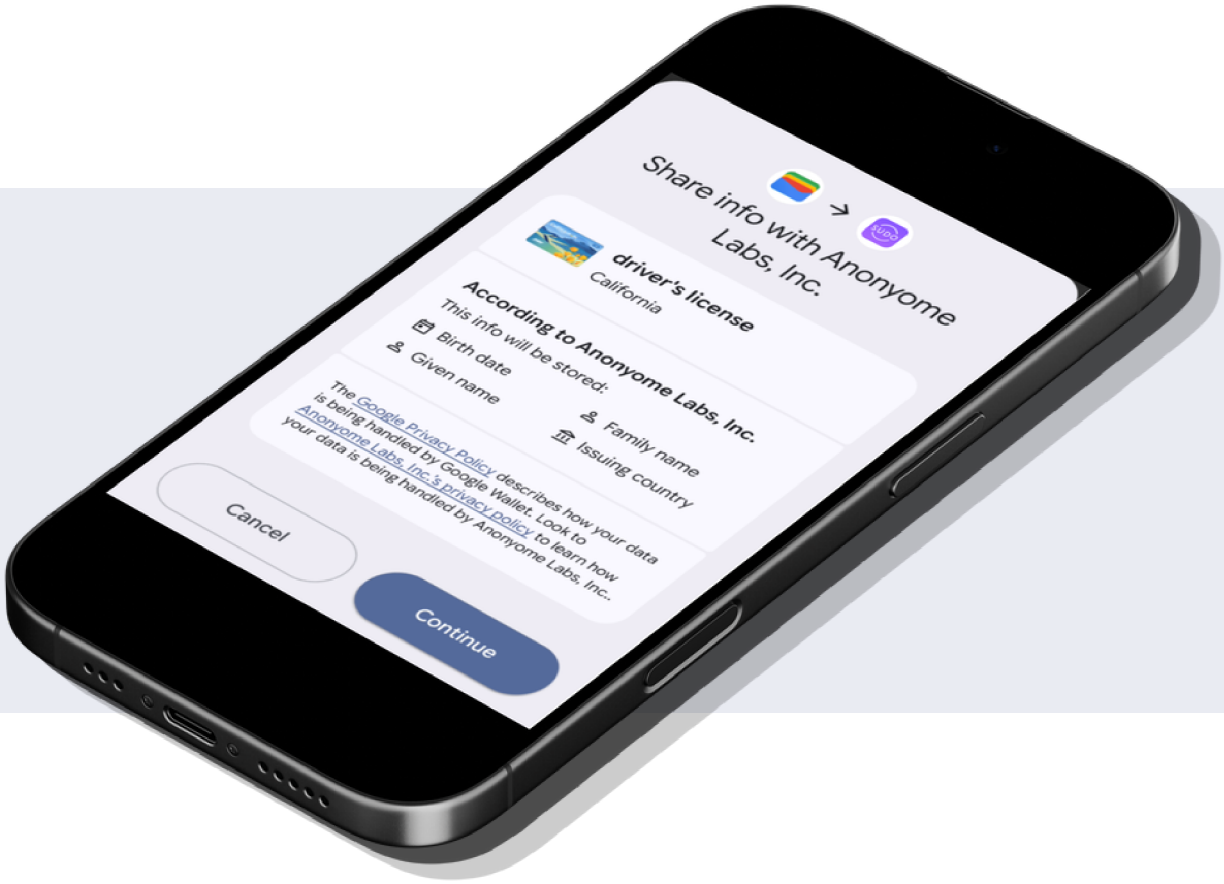
ANONYOME LABS®

White Paper

Enhancing the Anonyome Identity Verification Service to Support Digital Credentials from Apple and Google Wallets



Abstract

Identity Verification (IDV) enables a user to prove their legal identity. Current approaches involve the user providing personal information about their identity to the IDV system. This may involve the user manually entering data into an online form. In addition, the user can scan government identity documentation such as passports or driver licences. These approaches are error prone and tedious for the user. These methods are also vulnerable to impersonation attacks.

Conversely, using digital credentials to prove legal identity has significant advantages. The identity credentials are digitally signed, can be cryptographically verified, and support privacy enhancing selective disclosure. This paper explores how Anonyome's own Identity Verification Service (IDV Service) is being updated to support these new identity focused digital credentials from the Apple Wallet and Google Wallet. Users now have a streamlined, secure and privacy focused approach to Identity Verification.



Introduction

The Anonyome Identity Verification Service [1] allows a user to provide evidence of their legal identity. This is often essential for applications with regulatory compliance requirements, such as in the telephony, financial and health care sectors. One example of a regulatory requirement is Know Your Customer (KYC) which involves the application confirming the legal identity of a user before they are allowed access. Our own MySudo [2] private payment feature requires KYC and leverages the Anonyome Identity Verification Service.

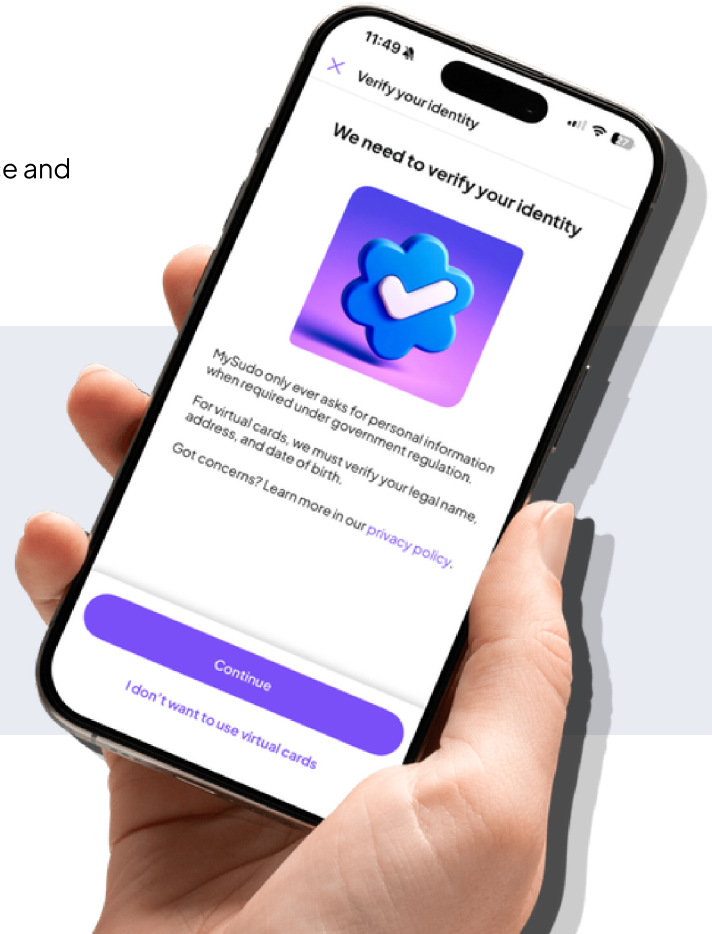
Traditionally, use of the Anonyome Identity Verification Service requires a user to manually enter their own personal data into a web form. Additionally, the user can scan government documents such as their Passport or Driver License. Optical Character Recognition (OCR) is then used to convert the images to text. In both cases, errors can be encountered. For example, the user can input the incorrect information or due to poor lighting or poor condition of a government document, the OCR provides incorrect information. This approach can be quite frustrating for the user, since a failed identity verification will require them to re-start the process.

With the recent introduction of electronic digital credentials in user mobile wallets (e.g., Apple Wallet, Google Wallet), the Anonyome IDV service can leverage these credentials to improve the overall user experience. For the user, it is much more user friendly to allow the user to present an electronic credential from a mobile wallet rather than requiring manual data entry or document scanning. The digital credential KYC method is error free since the data is electronically transferred and digitally signed.

For IDV service, digital credentials provide immediate cryptographic verification enabling the verifier to instantly determine:

- which issuer created the credential,
- that the credential data is unchanged,
- that the credential has been issued to this device and
- whether the credential has been revoked

In addition, digital credentials are privacy preserving as they support selective disclosure, which allows the user to only provide a required subset of the information contained within an identity credential. Digital credentials are also usually built according to international standards (e.g. ISO, IETF), which allows compatibility across systems and borders.



Digital Credential Specifications and Architectures Related to Identity Verification

There are several digital credential standards that are used for identity verification.

mDL and mDoc

Perhaps, the most universally accepted identity based digital credential is the Mobile Drivers License (mDL). The specifications behind the mDL have been developed over a number of years at the International Standards Organization (ISO) and have become almost universally accepted around the world (e.g., US, Canada, Australia, Japan, Korea, European Union).

There are two original specifications for mDL:

- ISO/IEC 18013-5:2021 Personal identification — ISO-compliant driving licence Part 5: Mobile driving licence (mDL) application [3]. This is the primary global standard that defines how mDLs work “in-person” without needing internet connectivity. Presentation is over NFC, Bluetooth Low Energy (BLE) or Wi-Fi Aware.

- ISO/IEC TS 18013-7:2025 Personal identification — ISO-compliant driving licence Part 7: Mobile driving licence (mDL) add-on functions [4]. This standard defines how an mDL is presented over the internet (e.g., to a website or remote verifier). This standard is typically chosen for online identity verification.

From this initial mDL work, these capabilities have been broadened to other identity credentials. ISO/IEC 23220-1:2023 Cards and security devices for personal identification — Building blocks for identity management via mobile devices [5] generalizes the foundations of 18013-5 into a series of documents with the aim to support any kind of authoritative digital document that can be carried in a mobile wallet and presented securely. ISO/IEC 23220 is often described as the evolution of the mobile driving licence work into a general-purpose digital credential framework.

The mDoc (Mobile Document) is the foundation credential type for these specifications. mDoc was originally defined in the ISO/IEC 18013-5 specification including how the credential is encoded (using Concise Binary Objective Representation; CBOR) and how it is signed (using CBOR Objective Signing and Encryption; COSE).

OpenID4VCI and OpenID4VP

OpenID for Verifiable Credential Issuance [6] (OpenID4VCI or OID4VCI) defines how a credential moves from the issuer to a wallet and is built on OAuth 2.0 and OpenID Connect patterns, which lower the barrier for existing identity systems to add credential issuance. The credential offers are made via URI or QR code for same-device or cross-device processing and support either pre-authorized-code and authorization-code flows, with proof-of-possession key binding and batch issuance. It supports W3C VC, SD-JWT and ISO/IEC 18013-5 mDoc formats.

OpenID for Verifiable Presentations [7] (OpenID4VP or OID4VP) defines how a credential moves from the wallet to a verifier. The wallet acts as both authorization server and resource server, the verifier sends an authorization-request-like message asking for specific claims via a Digital Credentials Query Language (DCQL) [7] query, the wallet gathers user consent, and then returns a signed response from the wallet proving ownership of one or more digital credentials. It supports W3C VC, SD-JWT and ISO/IEC 18013-5 mDoc formats. OpenID4VP works over both redirect-based (browser) and same-device/cross-device flows.

W3C Digital Credentials API

W3C Digital Credentials API (DC API) [8] is emerging as the browser-native interface for using digital credentials.

For credential presentation, a website calls the DC API (navigator.credentials.get()) with a protocol specific credential request. The operating system shows the user which wallets can satisfy it, the user selects one, approves the presentation, and the response is returned to the verifier to decrypt and validate. The API doesn't depend on which protocol the wallet and verifier have implemented internally. Today, two families of protocols are supported: OID4VP 1.0 variants distinguished by request-signing mode (carries SD-JWT VC and mDoc) and mDoc device retrieval (with a native mDoc binding from ISO/IEC TS 18013-7:2025 Annex C). Additional protocols can be added as the ecosystem evolves.

Similarly, a credential issuer may request credential issuance via the DC API (*navigator.credentials.create()*) request.

Apple announced support for DC API at WWDC 2025. Google announced support for DC API for Chrome 141 and above.

mDL Implementations

mDL is supported by several different implementations including native operating system mobile wallets and mobile application wallets. The US Transport Security Administration (TSA) [9] supports presentation of the mDL from Apple Wallets, Google Wallets, as well as, from a number of Department of Motor Vehicle Wallet mobile applications.

mDL Support in Apple (iOS) Wallet

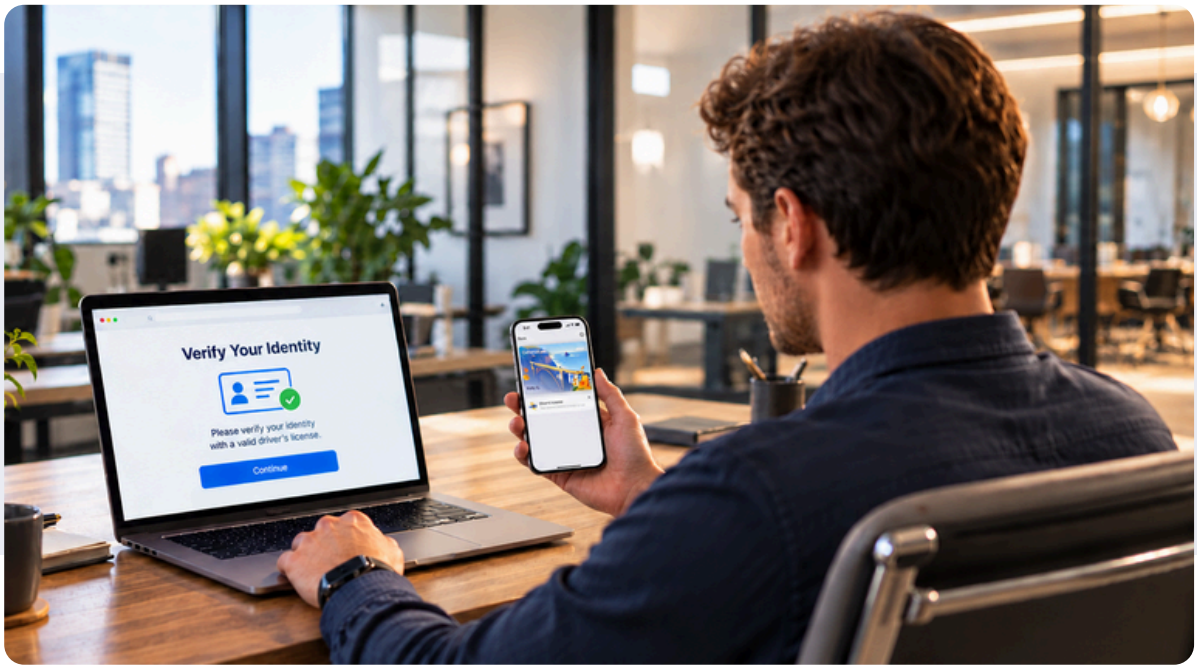
The Apple Wallet now provides support for mDL. Table 1 shows the Apple Wallet technical implementation details.

Apple supports mDL using ISO/IEC 18013-5 (mDoc) as the underlying data model. The Apple mDoc supports device binding to prevent cloning of the mDL to another device.

A user will initiate the issuance flow by opening their Apple Wallet and selecting “Driver’s License or State ID”. The user then scans the front and back of their physical driver license and performs a liveness biometric check. Apple will help facilitate the transfer of information to the DMV, but it is up to the DMV to determine whether they will issue the mDL license. This protocol is entirely Apple proprietary.

For in-person mDL presentation, a secure device-to-device connection is established over NFC or through a QR code, similar in feel to tapping for mobile payments. Once engaged, data is transferred over a secure channel using Bluetooth Low Energy (BLE), NFC, or Wi-Fi Aware.





For online mDL presentation, there are two alternatives:

- In-App: Apple uses Verify with Wallet API which is a proprietary implementation
- Web/Browser: Apple uses Verify with Wallet on the Web, which supports the 18013-7:2025 presentation protocol (Annex C). This builds on the 18013-5 device-retrieval framework, but uses the W3C Digital Credentials API to establish the session and request credentials from the holder's wallet. Annex C currently supports only the mDoc format.

Credential Format	**ISO/IEC 18013-5 mdoc**
Issuance Protocol	**Proprietary Apple Protocol**
Verifier Protocol (In-Person)	**NFC/QR device engagement → BLE/NFC/Wi-Fi Aware transfer**
Verifier Protocol (Online)	**(In-App) Verify with Wallet API, Proprietary (Web/Browser) Verify with Wallet on the Web, ISO/IEC TS 18013-7:2025 Annex C via W3C Digital Credentials API**

Table 1: Apple Wallet mDL Technical Implementation

mDL Support in the Google (Android) Wallet

The Google Wallet also provides support for mDL. Table 2 shows the Google Wallet technical implementation details.

Google supports the mDL as defined by the data model being the ISO/IEC 18013-5 defined mDoc. The Google mDoc supports device binding to prevent cloning of an mDL to another device.

To add a license, the user opens the Google Wallet, taps Add to Wallet, photographs the front and back of the driver licence, and records a short selfie video. A photo from that video is submitted to the DMV issuer for verification. Google will help facilitate the transfer of information to the DMV. However, the DMV determines whether they issue the mDL license. This protocol is Google proprietary.

For in-person mDL presentation, a secure device-to-device connection is established over NFC or through a QR code, similar in feel to tapping for mobile payments. Once engaged, data is transferred over a secure channel using Bluetooth Low Energy (BLE), NFC, or Wi-Fi Aware.

For online presentation, the Google Wallet requires requests to be formatted using 18013-7:2025 Annex B (presentation of mDoc using OID4VP). This builds on the 18013-5 device-retrieval framework but uses the W3C Digital Credentials API to establish the session and request credentials from the holder's wallet. It supports mDoc-based digital IDs with a DCQL query object specifying requested credentials and attributes.

Credential Format	**ISO/IEC 18013-5 mDoc**
Issuance Protocol	**Proprietary Google Protocol**
Verifier Protocol (In-Person)	**NFC/QR device engagement → BLE/NFC/Wi-Fi Aware transfer**
Verifier Protocol (Online)	**W3C DC API / Credential Manager, OpenID4VP as default transport**

Table 2: Google Wallet mDL Technical Implementation

mDL Support in DMV wallet apps

There are several DMV based wallet apps that support mDL. The most well-known is the California DMV Wallet [10] that supports mDL. Table 3 shows the California DMV Wallet technical implementation details.

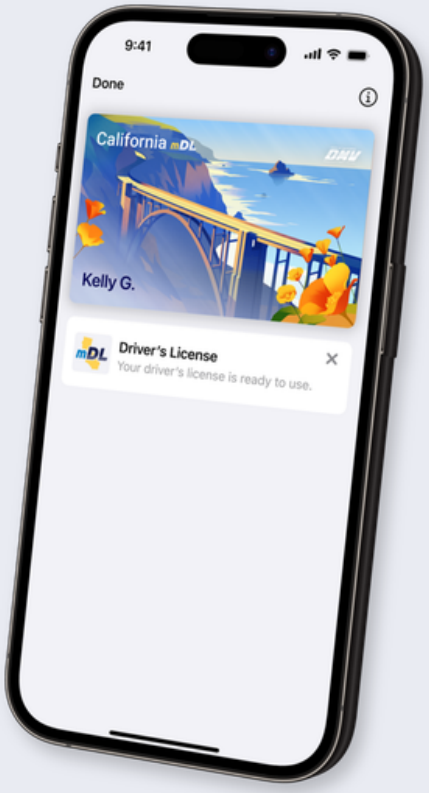
The California DMV Wallet supports the mDL using the ISO/IEC 18013-5 defined mDoc. The California DMV Wallet mDoc supports device binding to prevent cloning of the mDL to another device. The wallet also supports the W3C VCDM verifiable credential for greater interoperability.

As a real technical distinction from Apple and Google, the California DMV directly co-hosted OpenID Foundation hackathons in October and November 2024 specifically to test OpenID4VCI issuance flows. One of the tested configurations was explicitly "OpenID4VCI with mDoc, Custom URI initiated, no client authentication."

For in-person mDL presentations, a secure device-to-device connection is established over NFC or through a QR code. Once engaged, data is transferred over a secure channel using Bluetooth Low Energy (BLE), NFC, or Wi-Fi Aware.

The California DMV app supports two modes of online presentation:

- OpenID4VP directly (via a QR Code scan or deeplink to the wallet)
- DC-API, California DMV app is a registered credential provider

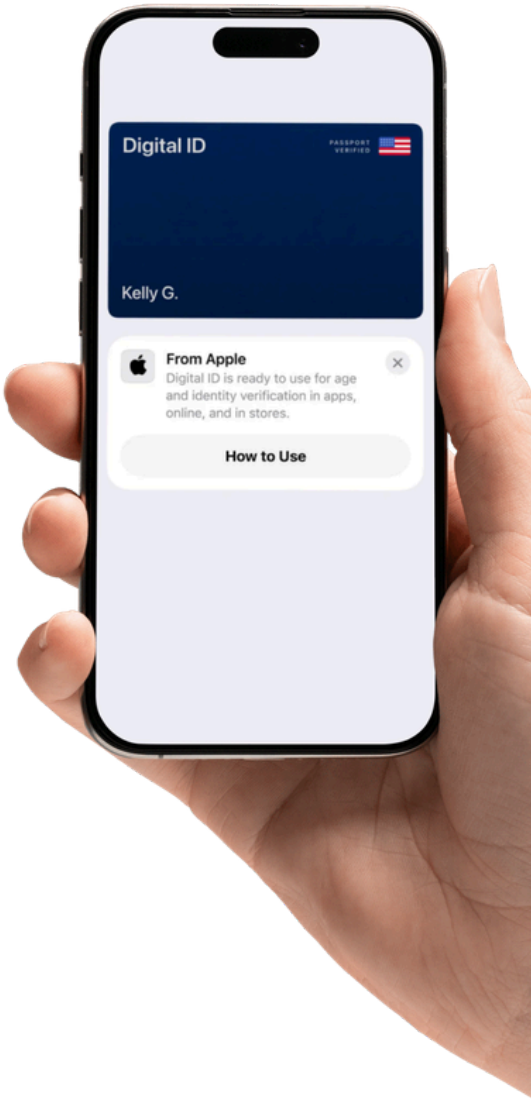


Credential Format	**ISO/IEC 18013-5 mdoc and W3C VC DM**
Issuance Protocol	**Piloting OpenID4VCI**
Verifier Protocol (In-Person)	**NFC/QR device engagement → BLE/NFC/Wi-Fi Aware transfer**
Verifier Protocol (Online)	**OpenID4VP directly (via a QR Code scan or deeplink to the wallet) or DC-API, CA-DMV app is a registered credential provider.**

Table 3: California DMV Wallet app mDL Technical Implementation

Alternative Identity Related Digital Credentials

Not everyone has a driver license and Apple and Google have worked to support alternative Identity based digital credentials.



Apple Digital ID

Apple's Digital Identity Credential (Digital ID) is a new capability released in iOS 26 [11].

The Digital ID is created from information supplied from the user's passport. The user on-boards for this credential using their Apple wallet app interface. The user scans their passport image and allows the Apple wallet app to have direct access to the information stored in the NFC chip embedded in the passport. Additionally, a number of liveness test functions must be completed by the user. At the end of the process, a cryptographically protected Digital ID (in mDoc format) representing the data from the passport is created in the user's Apple wallet.

The TSA [9] is already conducting trials using the Apple Digital ID at airports in order to allow for secure identification of users that want an alternative to using mDL. The user will authenticate to their mobile device using the biometric capability in their iOS devices, which also unlocks their wallet app enabling them to present their Apple Digital ID.



Google ID Pass

Google's ID Pass [12] is very similar in design to the Apple Digital ID. A user will use the Google Wallet app to scan their passport image, access the passport information on the passport's NFC chip, and provide proof of liveness. The outcome is a new Google ID Pass credential (in mDoc format) in their wallet. This credential is also accepted by the TSA [9] and requires mobile device biometrics to unlock the wallet app which presents the Google ID Pass.

Implementation of the Identity Verification Service Enhancements

As described in previous sections, a user has several different options when deciding which identity based digital credential to use for identity verification. One way to quickly understand the options is to review the TSA's own web site [9], which describes several allowed options. The enhancement to the Anonyome IDV service will initially support presenting:

- an mDL from Apple Wallet
- an Apple Digital ID from Apple Wallet
- an mDL from Google Wallet
- a Google ID Pass from Google Wallet

In all cases, the Anonyome IDV service becomes a Relying Party for the digital credentials.

Becoming a Relying Party for Google Wallet Digital Credentials

Phase 1: Registration with Google

To become a Relying Party that can request a digital credential from the Google Wallet, registration with Google is required [13]. Figure 1 shows the process flow for registering as a Relying Party for the Google Wallet.

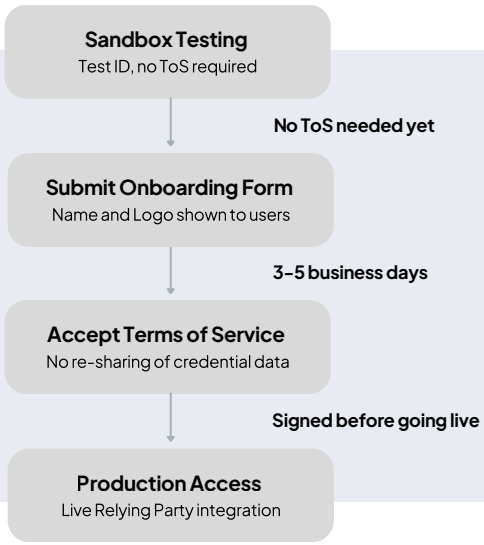


Figure 1: Google Registration for a Relying Party

Registration is a lightweight, staged process and development can begin immediately in the Sandbox Environment using a Test ID.

When moving to production, a Relying Party must submit the Onboarding Form. This form requires a certificate signing request (CSR) that contains the Relying Party's public key. The Product Name and Logo supplied during this process will appear on the mobile device's user-facing consent screen so end users can identify who is requesting their data. In the form, the Relying Party needs to disclose what data elements (attributes) from credential they request and for what purpose, how this data is processed, stored and for how long. Also included in the form are links to the Relying Party's Privacy Policy and Terms of Service. Before going live, the Relying Party must formally accept Google's Terms of Service, which include a strict prohibition on re-sharing credential data with third parties.

As an output of this registration process, Google provides the Relying Party with a signed certificate which verifiably indicates Google's authorization for the Relying Party to make digital credential requests to the Google Wallet.

Organizations managing trust relationships for multiple downstream clients have a separate path, onboarding as a Verifier Registrar rather than a single Relying Party. Although this does allow for different branding (product name and logo) for different Relying Party applications, there is additional configuration and on-going maintenance required.

Phase 2: Relying Party requesting a Digital Credential

Figure 2 shows the flows for the Relying Party requesting the presentation of a digital credential from the Google Wallet. In this example, the request begins at the MySudo application, which in this case is requiring identity verification for our private payments feature. From MySudo, the Relying Party requests presentation of a digital credential, to the Android operating system credential manager, then to the Google Wallet and back again.

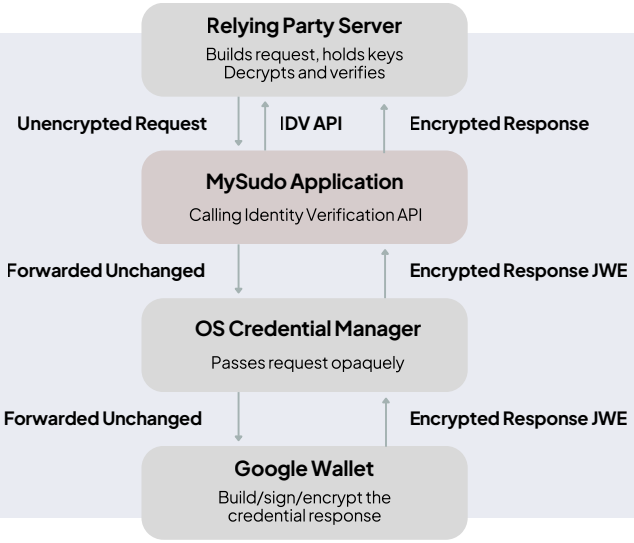


Figure 2: Google Relying Party Request/Response flows

Request Message

The Relying Party constructs the request server-side in JSON containing a DCQL Query that requests an mDoc format credential, of some particular document type (such as Google ID Pass or mDL), and a claims array describing the attributes to be presented. The request can be signed by the Relying Party using the private key matching the previously registered public key (from the Phase 1 Google registration process). Requests can target one specific credential or offer alternatives (e.g., mDL or ID Pass) via credential sets. Requests are unencrypted. If the Relying Party wants the wallet response to be encrypted, it passes a new public key in the request.

On Android, the request is passed opaquely through Credential Manager to whichever wallet matches the request. The wallet receiving the request can perform its own validation of the incoming request according to its own policy. In the case of Google Wallet handling the request, validation is performed to ensure the request is correctly signed from a registered Relying Party, as indicated by the trusted Google-signed certificate chain included in the request. Additionally, Google Wallet after validating the signature and trust chain will display the branding assets of the Relying Party (as registered during Phase 1).

Reply Message

The user's chosen wallet returns an encrypted response, which the Relying Party decrypts server-side using its private key.

Validation is the critical step. The Relying Party must be able to validate the response according to ISO/IEC 18013-5:2021 clause 9. The first step is to extract the issuer's signing certificate chain and validate it against the trusted IACA root certificates. The digests and signatures across the response are then verified for correctness.

Supporting Multiple Wallets

As shown in Figure 3., the credential manager does not decide which wallets can fulfill the response, this determination is made by the wallets themselves.

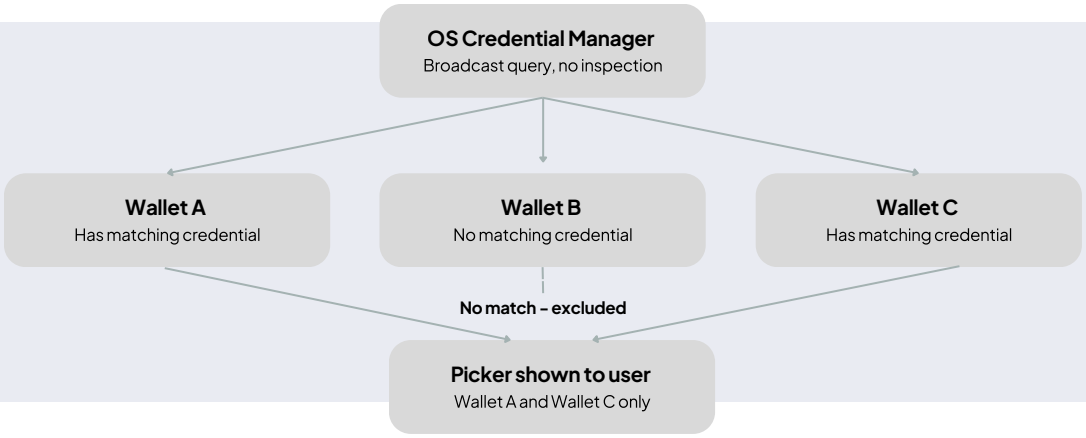


Figure 3: Credential Manager fan-out to the wallets

The Credential Manager passes the request message along to the available digital wallets without examining its contents. The Credential Manager doesn't inspect the DCQL query or know what the document type is or claims are. It is format-agnostic by design, which is consistent with the DC API's general design of staying protocol-neutral.

The actual matching intelligence lives in each wallet. Each wallet is responsible for parsing the request to understand the identity requested and determining which of its stored digital credentials, if any, satisfy the request. Every wallet installed on the device independently evaluates the DCQL query against its own stored credentials and reports back whether it can satisfy it.

Credential Manager's actual job is orchestrating that fan-out and then presenting a picker which shows the user which wallets can satisfy the request, and the user selects a source explicitly.

Becoming a Relying Party for Apple Wallet Digital Credentials

Phase 1: Registration with Apple

To become a Relying Party that can request a digital credential from the Apple Wallet, registration with Apple is required [14]. Figure 4 shows the process flow for registering as a Relying Party for the Apple Wallet.

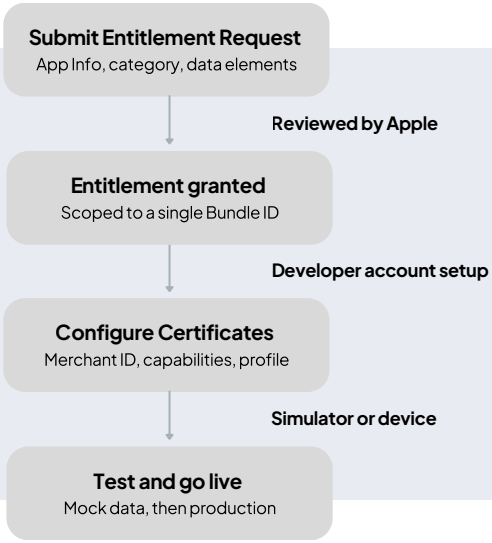


Figure 4: Apple Registration for a Relying Party

Apple controls access to this process at the entitlement level. It is possible to start testing the Apple Verify with Wallet API in development before having anything granted by using Apple's developer profiles on an iOS device or simulator.

Two conditions gate entry for production deployment:

- **Consistency requirement:** the app must require an equivalent age or identity verification process for every user obtaining the same goods or services in each jurisdiction.
- **Second, a category restriction:** the app must fall into one of twelve approved categories i.e. Access (physical security), Air Travel, Alcohol Purchase, Car Rental, Financial Services, Gig Economy, Government Services, Healthcare, Hospitality, Insurance, Scooter Rentals, or Ticketing.

The entitlement is scoped to a single app binary and is not an organizational wide grant.

The request needs the app name, bundle ID, App Store/TestFlight URL, a description of the good or service requiring verification, and screenshots of your existing verification flow for users without a wallet-based ID.

Apple also requires the specific data elements intended to be requested, drawn from two buckets: age-verification data (age-over-N flag, age in years, issuing authority, photo) or identity-verification data (name, address, DOB, physical descriptors, document number/dates, driving privileges, Real ID compliance flag, and others). Requestors must also supply privacy policy URLs.

Once granted, the administrator inputs a Merchant ID, generates an Identity Access Certificate, add the "In App Identity Presentment" capability to the App ID scoped to the requested document types and elements, and adds the separate merchant-identifier capability.

Phase 2: Relying Party requesting a Digital Credential

Functionally, the Apple flow is similar to the Google flow. The difference is transport (native PassKit API vs. DC API/OpenID4VP-carried DCQL). Figure 5 shows the process flow for the Relying Party requesting the presentation of a digital credential from the Apple Wallet. In this example, the request begins at the MySudo application, which in this case is requiring identity verification for our private payments feature. MySudo (on behalf of the Relying Party Server) requests presentation of a digital credential, using the Apple PassKit. The user approves and the Apple Wallet creates the response. Apple provides standard PassKit buttons ("Verify with Apple Wallet," "Verify Age/Identity with Apple Wallet") for a consistent system-level prompt.

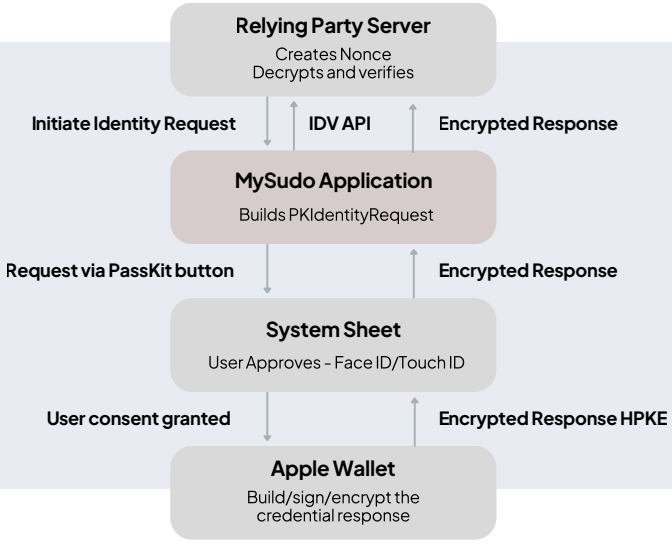


Figure 5: Apple Relying Party Request/Response flows

Request Message

The MySudo app builds an object on the client, which specifies the identity descriptor (e.g., U.S. for mDLs or for national IDs/passports) and the specific elements to request. The request is limited to the specific entitlement granted, since the request can't ask for elements beyond what Apple has approved.

The merchantIdentifier indicates the encryption certificate to which the API response will be encrypted. A nonce is created (usually by the Relying Party server) which will be tied to the response received. This is an important security feature used to prevent replays of a response and to bind it to a specific user session.

Reply Message

If the user approves, the wallet produces a payload encrypted to the merchant identifier's encryption certificate. The Relying Party app never sees plaintext identity data, only ciphertext, which it hands off to its own server.

The Relying Party server decrypts using the private key of the Identity Access Certificate created against its Merchant ID. It next verifies the device signature against the session transcript, per ISO/IEC 18013-5's device-authentication model. This confirms the response came from the specific device/session the request targeted, not a replay. It then verifies the issuer signature against the relevant IACA root certificate such as the state DMV's, the Government of Japan's, or Apple's own (for the passport-based Digital ID path), depending on which issuing authority signed the credential.

Supporting Multiple Wallets

One key difference to the Google process is that there is no fan-out step. Google's Credential Manager broadcasts to every installed wallet and lets each one self-select. Apple's Verify with Wallet API flow has exactly one wallet in the loop by construction, so the "choice" moment that exists on Android simply doesn't occur on iOS. The system sheet here is a consent prompt, not a selector.

Apple has also extended its capabilities with Verify with Wallet on the Web [15]. This brings DC-API support to iOS (rather than just via the Apple Wallet API) where wallet apps beyond Apple Wallet can respond to credential requests, as with Android. However, unlike Android, this capability can only be invoked from the context of a browser, such as Safari, and not from in-app like can be done with Verify with Wallet API. It is therefore out of scope for this paper as it is less relevant to Anonyome Identity Verification process through native mobile apps and SDKs.

References

[1] The Anonyome Identity Verification Service, https://docs.sudoplatform.com/guides/identity-verification

[2] MySudo Application, https://anonyome.com/individuals/mysudo/

[3] ISO/IEC 18013-5:2021 Personal identification — ISO-compliant driving licence Part 5: Mobile driving licence (mDL) application, https://www.iso.org/standard/69084.html

[4] ISO/IEC TS 18013-7:2025 Personal identification — ISO-compliant driving licence Part 7: Mobile driving licence (mDL) add-on functions, https://www.iso.org/standard/91154.html

[5] ISO/IEC 23220-1:2023, Cards and security devices for personal identification — Building blocks for identity management via mobile devices, https://www.iso.org/standard/74910.html

[6] OpenID for Verifiable Credential Issuance (OID4VCI) 1.1 Working Group Draft, https://openid.github.io/OpenID4VCI/openid-4-verifiable-credential-issuance-1_1-wg-draft.html

[7] OpenID for Verifiable Presentations (OID4VP) 1.0, https://openid.net/specs/openid-4-verifiable-presentations-1_0.html

[8] Digital Credentials API Working Draft 17 April 2026, https://www.w3.org/TR/digital-credentials/

[9] Participating States and Eligible Digital IDs, https://www.tsa.gov/digital-id/participating-states

[10] California DMV Wallet, https://www.dmv.ca.gov/portal/ca-dmv-wallet/

[11] Apple introduces Digital ID, a new way to create and present an ID in Apple Wallet, https://www.apple.com/newsroom/2025/11/apple-introduces-digital-id-a-new-way-to-create-and-present-an-id-in-apple-wallet/

[12] New Google Wallet features for travelers and commuters, https://blog.google/products-and-platforms/platforms/google-pay/google-wallet-update-september-2024

[13] Online Acceptance of Digital Credentials, https://developers.google.com/wallet/identity/verify/accepting-ids-from-wallet-online#specific-credential

[14] Get started with the Verify with Wallet API, https://developer.apple.com/wallet/get-started-with-verify-with-wallet/

[15] Verify with Wallet on the Web FAQs, https://support.apple.com/en-au/guide/business/abcbb063f3a2/1/web/1