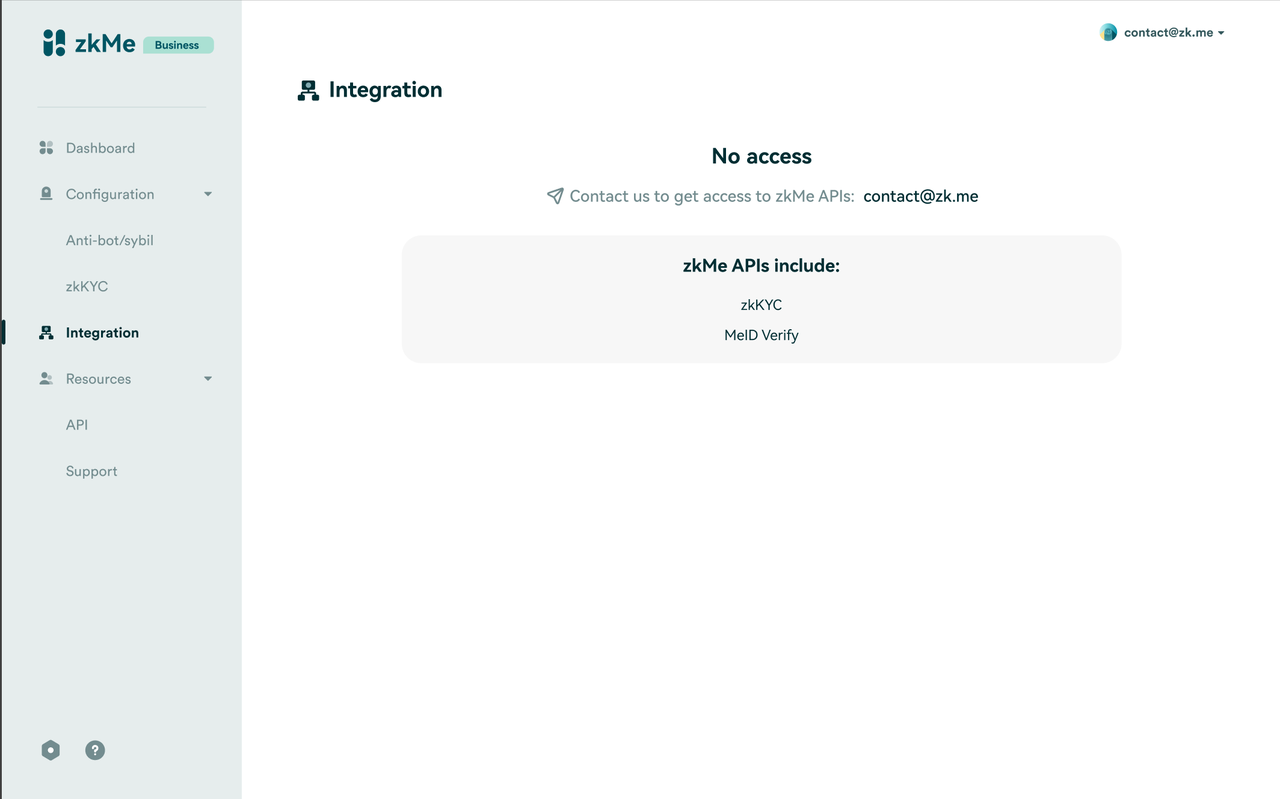
**zkMe Dashboard**

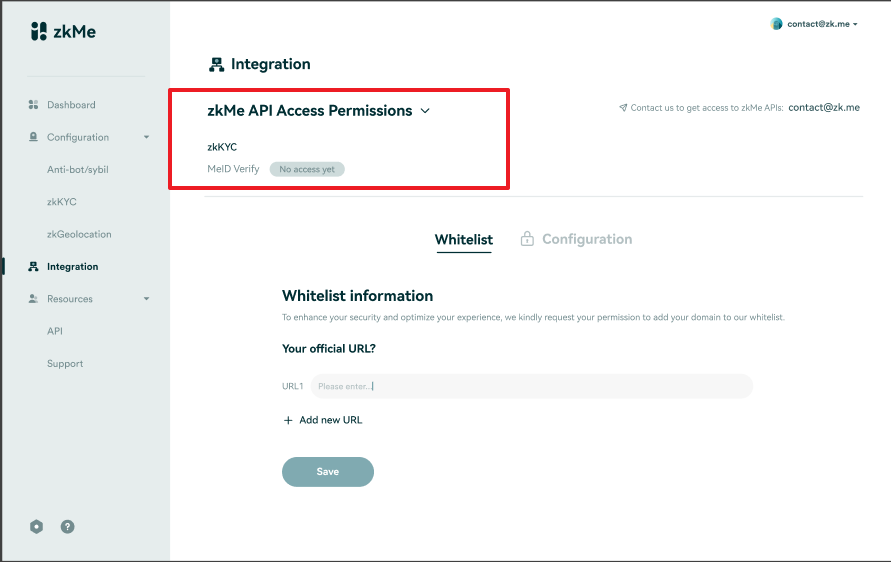
**zkMe dashboard Integration**

As a new customer who first logged into the zkMe dashboard by default has no zkMe permissions. You can get access to zkMe APIs by contacting contact@zk.me.



If you already have permission for a particular service, you will see the corresponding account configuration information. Fill in the relevant details to obtain a dapp account for each public chain, which is used to store your users' authorization information.

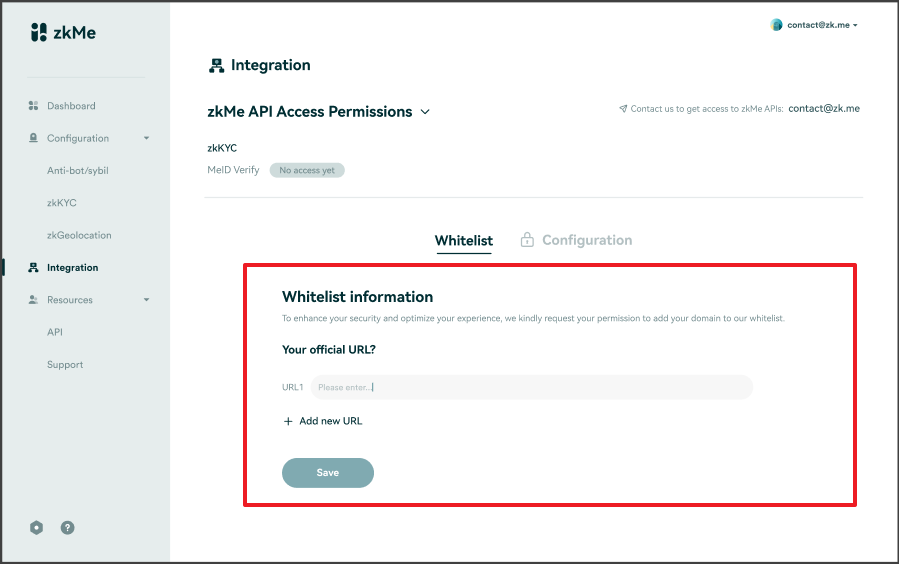
They need to go to the **Integration** part to complete the whitelist and account configuration before obtaining the corresponding zkMe verification permission.



**Whitelist setting**

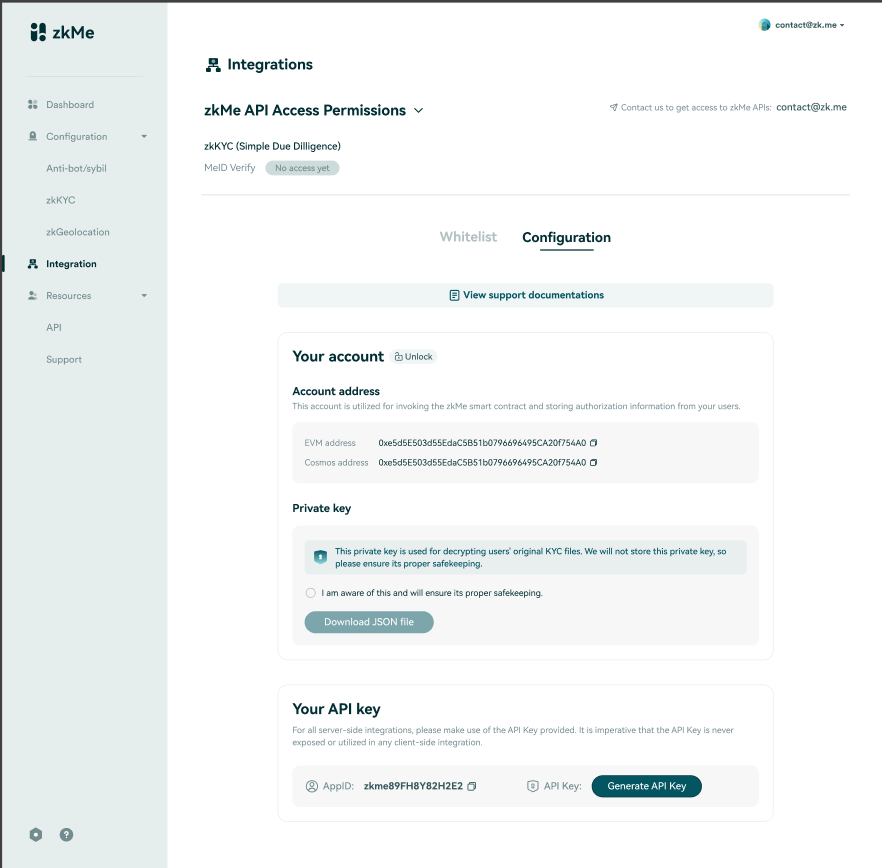
In this section, enter the URL of the verification service you wish to integrate. If you have multiple URLs, click on "**Add new URL**" to add a new input box. At this stage, we do not impose any restrictions on the number of addresses.

Please ensure that you fill in at least one valid URL address. After completion, click "**Save**", and you can then proceed to retrieve your dapp account information.



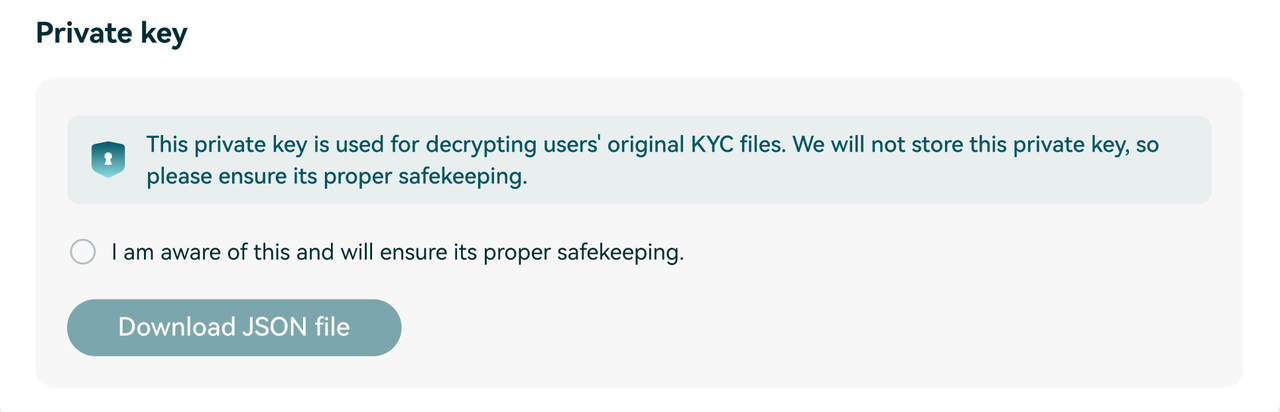
On this page, all types of public chains currently supported by zkMe will be displayed (currently: EVM and Cosmos). We will automatically generate a dapp account corresponding to each type of public chain for you and produce the corresponding threshold signature private key.

**P.S.** When integrating, select the corresponding smart contract based on the public chain you wish to connect to, and provide the relevant dapp account.



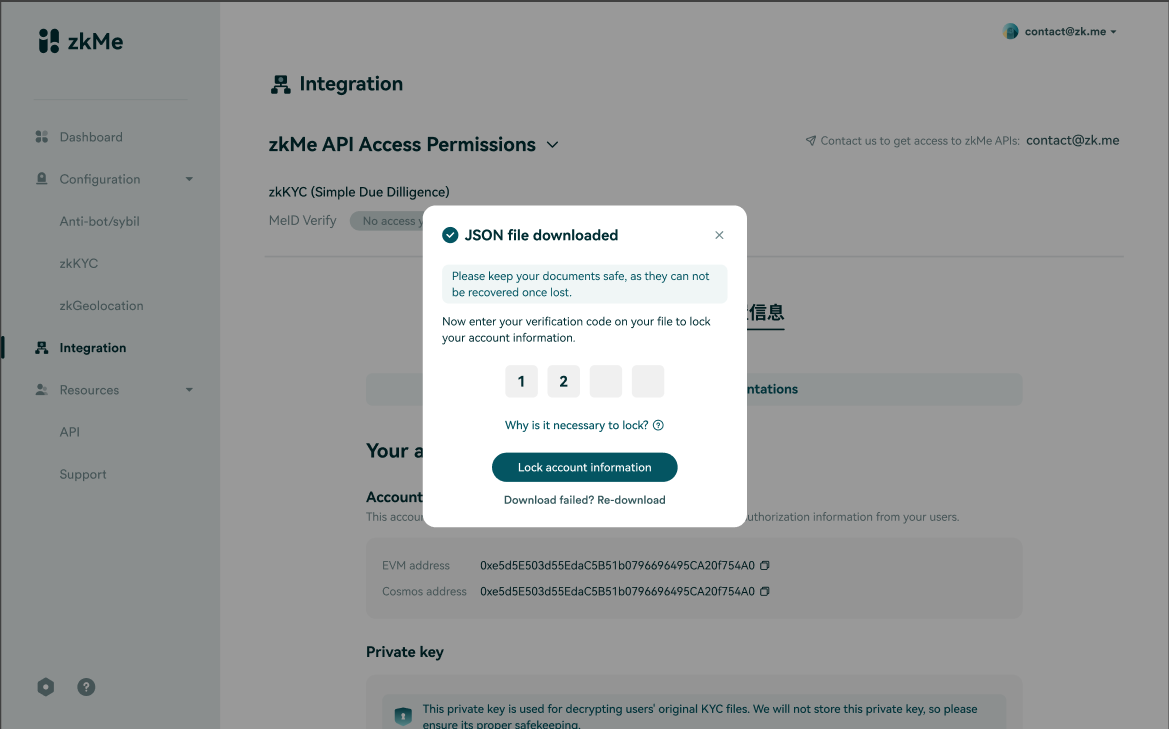
In the "Private key" section, please read the precautions regarding private keys, and check the awareness box, and download your private key JSON document.

Please note: The private key is generated once and zkMe will not save any of your private key information.



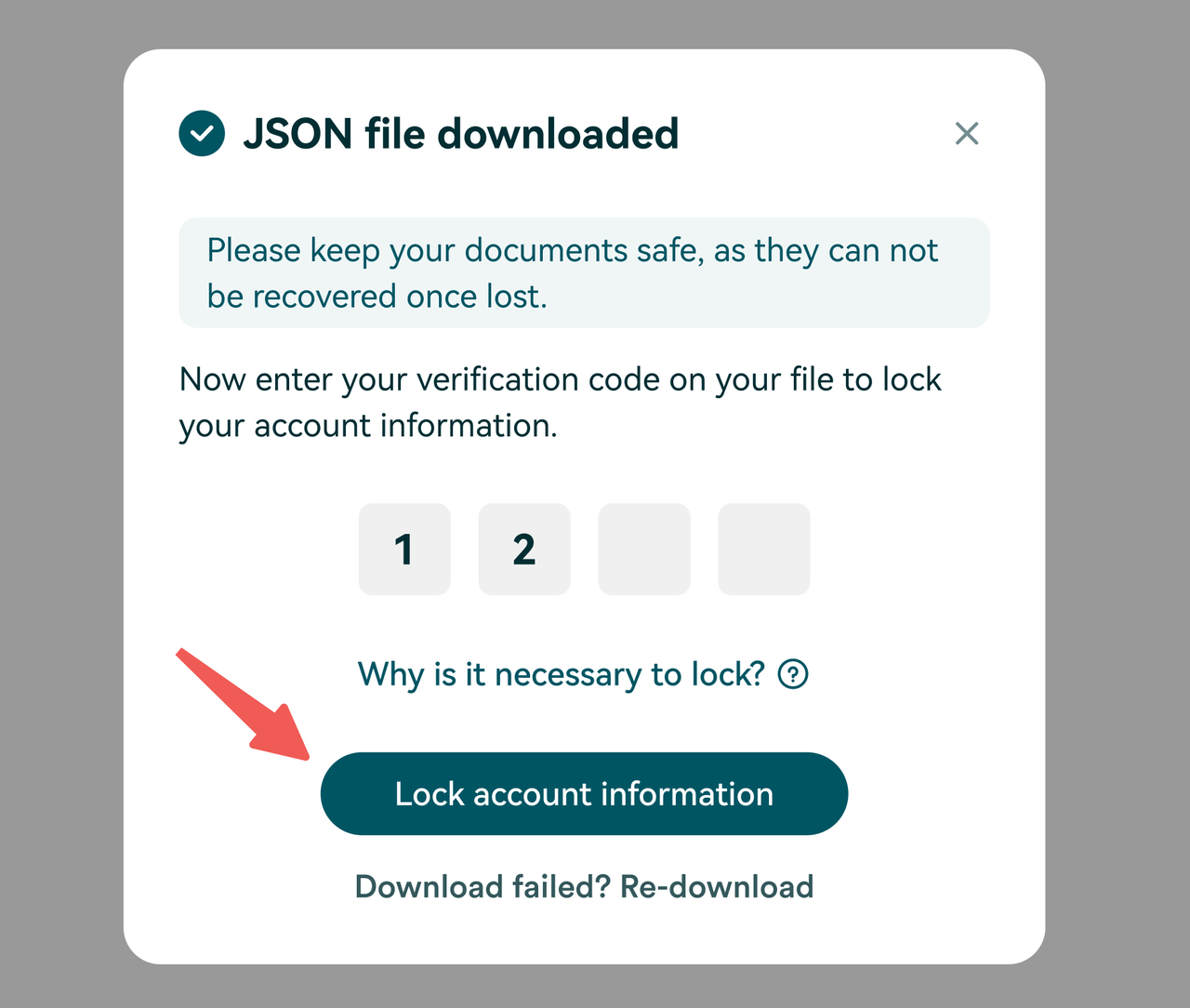
A verification code in the document should be provided to confirm whether you have successfully downloaded the private key JSON document, which is usually a four-digit number.

Please open the document to view the verification code and correctly enter it in the pop-up window to confirm.



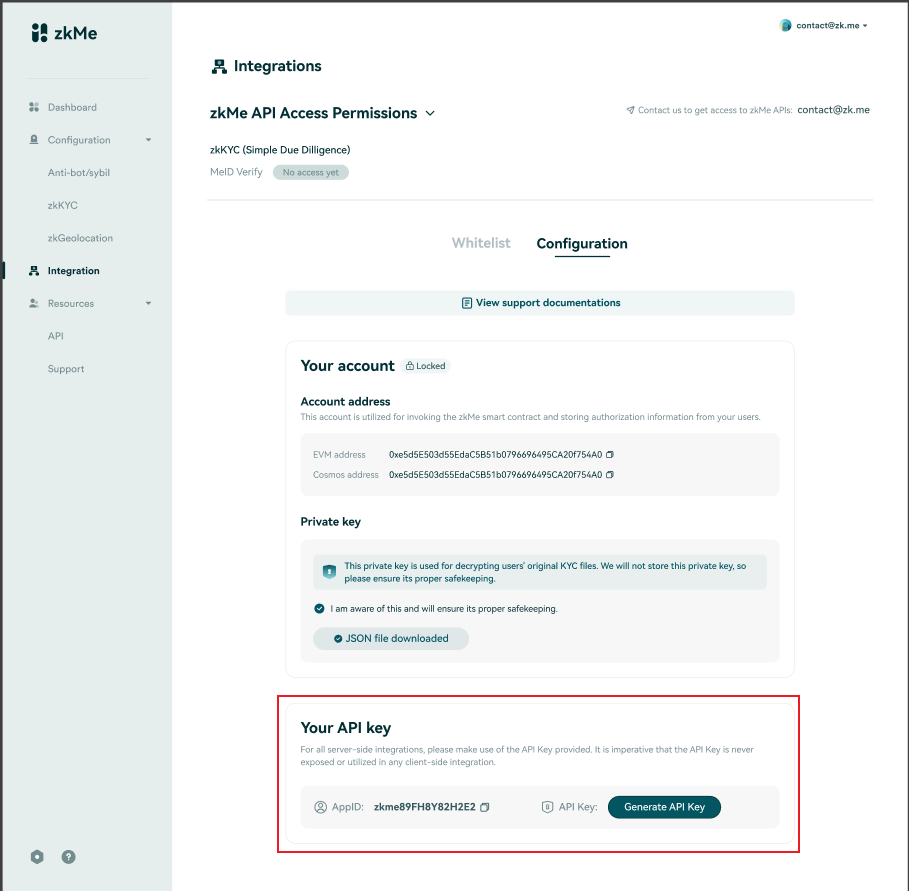
After confirming that you have properly saved the JSON document, please click the 'Lock' button to secure your dapp account information.

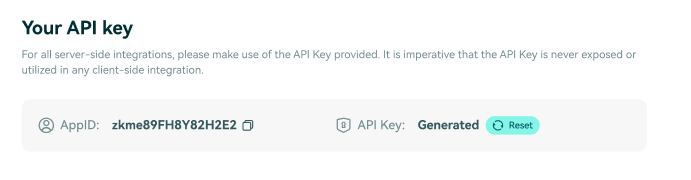
Please note: If you do not lock your account information, the next time you enter this module, your account details will change, and any previously integrated services will become invalid.



In addition to obtaining the dapp account, you also need to acquire the account's API key. The API key will be used to verify your permissions in the backend when your users launch the widget for KYC, ensuring that you have the relevant service permissions.

The API key can be retrieved multiple times, but once a new key is acquired, all services integrated with the old key will become invalid. Only after replacing with the new key can your users continue to use zkMe's verification services.





# zkMe Smart Contracts

Overview of the Smart Contracts (SC) developed by zkMe for the processing of the zkMe Network.

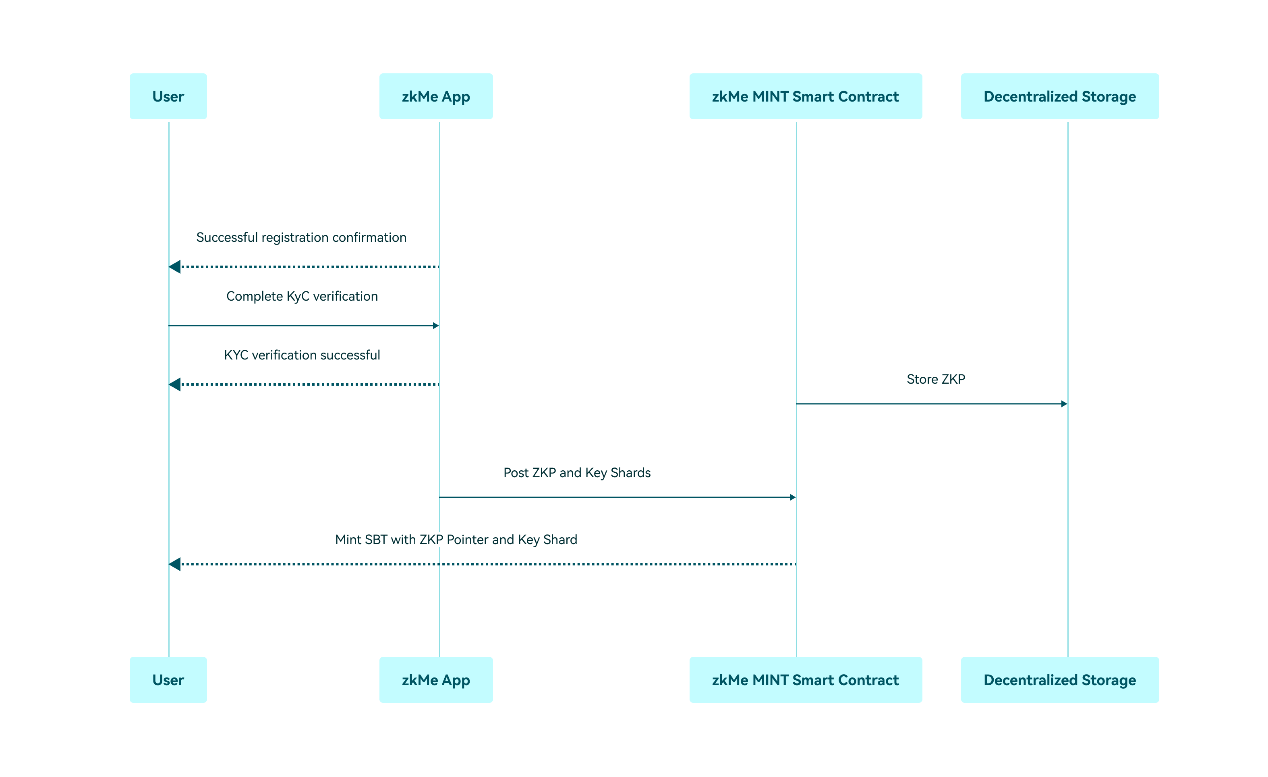
## Smart Contract Overview

To facilitate decentralized verification, zkMe developed a suite of smart contracts, enabling the protocol and Verifiers to process verifications autonomously. zkMe Mint and Delegate smart contracts mint the original and delegate copies of the DID onto the Holder's wallets. zkMe Verify and Certify smart contracts allow Verifiers to interact with the zkMe network and request minting of a special SBT copy for legal data access requirements if necessary. All functionalities available through the zkMe SCs are also available on zkMe APIs for non-web3 native Verifiers.

### zkMe Mint

The objective for the zkMe Mint smart contract is to anchor the verified credentials (and their anonymized presentations) on-chain.

The following sequence diagram shows the process of registering an SSI wallet, completing KYC verification, and minting an SBT token. The three participants involved are Holder, zkMe App (SSI Wallet), and Polygon (MATIC). This SC for minting SBT is deployed on the MATIC blockchain. The process starts with the Holder requesting to register an SSI wallet through the zkMe App. Once the wallet is created, the Holder presents their credentials to the zkMe App. The zkMe App generates the relevant ZKP and triggers the minting request to the zkMe Mint Polygon smart contract. The zkMe Mint SC receives the location pointers for the ZKP, minting an SBT asset directly to the Holder SSI wallet, The zkMe SBT, which contains the holder's DIDs, a key share, and most importantly the pointer to the verified ZKP. This process ensures that the Holder is able to securely own their Identity on-chain.



zkMe Mint address (Polygon Mumbai Testnet): **0xA0D0bBEC266a4F02aBeE4f7F5b7F20751eC1deba**

See:

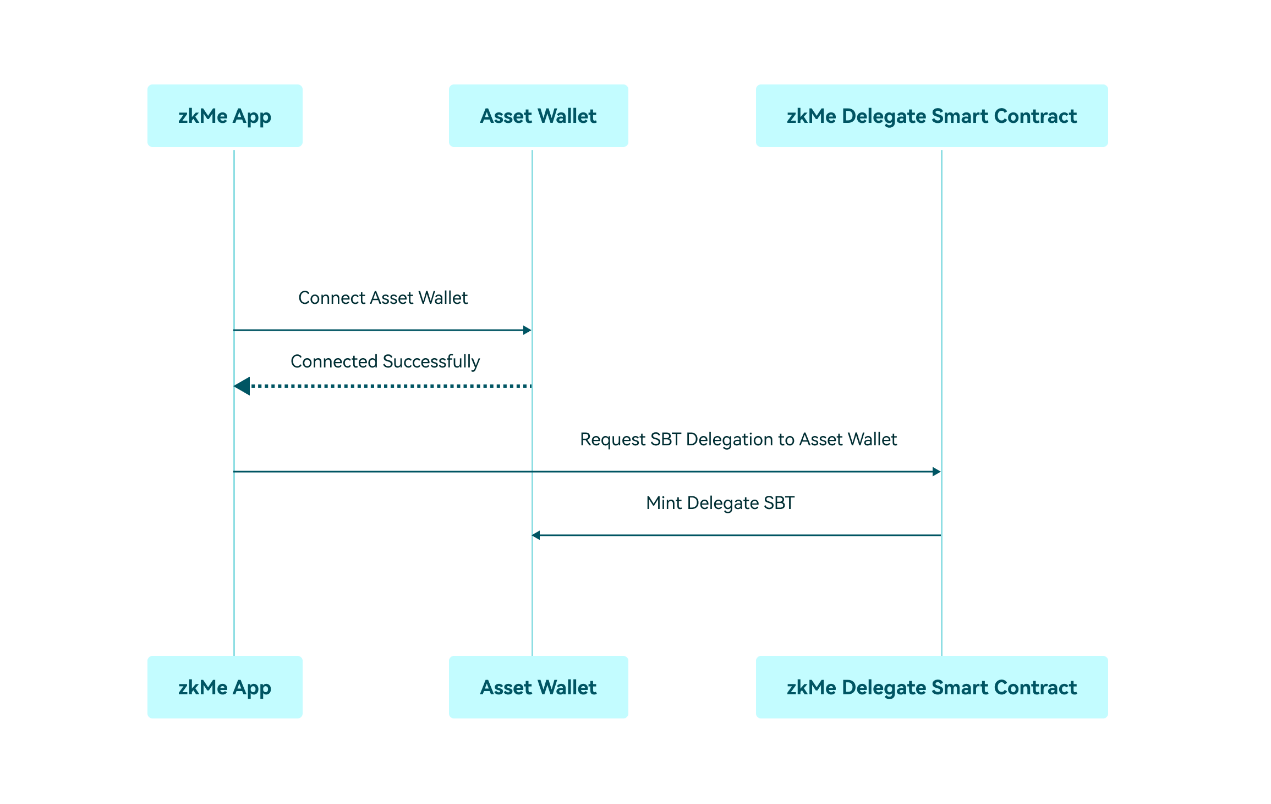
[TransparentUpgradeableProxy | Address 0xA0D0bBEC266a4F02aBeE4f7F5b7F20751eC1deba | PolygonScan](https://mumbai.polygonscan.com/address/0xA0D0bBEC266a4F02aBeE4f7F5b7F20751eC1deba" \l "code)

[Polygon (MATIC) Blockchain Explorer](https://mumbai.polygonscan.com/address/0xA0D0bBEC266a4F02aBeE4f7F5b7F20751eC1deba" \l "code)

### zkMe Delegate

The goal of the zkMe Delegate Smart Contract is making copies of the user’s on-chain Identities available on the chain ecosystems in which the user is active in.

The zkMe Delegate SC comes into play when a Holder wishes to perform verifications for dApps across chain ecosystems. Holders need to first connect their asset wallet to the zkMe App and sign a transaction requesting a delegate copy of SBT. The zkMe infrastructure and zkMe Delegate SC complete the cross-chain data transfer to the Holder's connected asset wallet and issue a delegate copy of SBT. Currently, zkMe Delegate supports the ETH, MATIC and BNB chains. Support for additional EVM-compatible chains is achievable with minimal efforts.

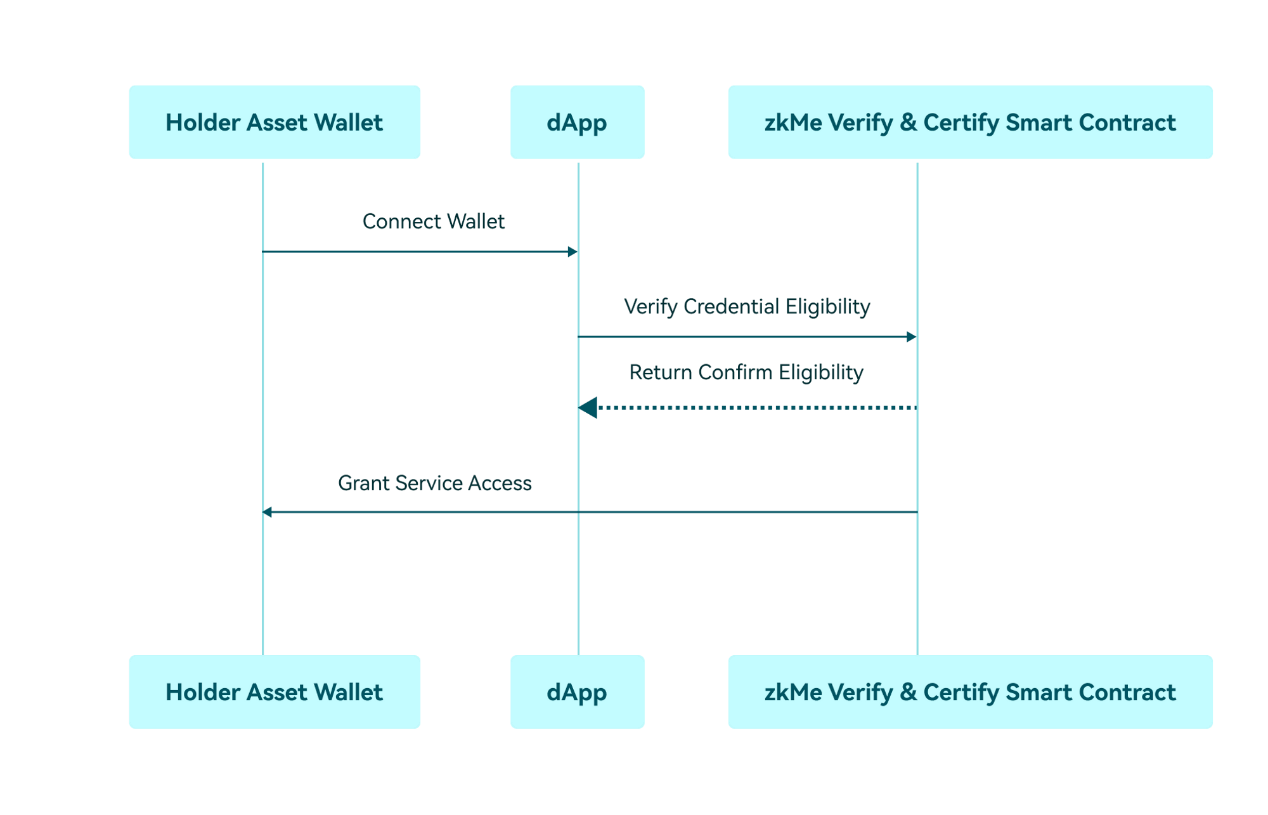


| zkMe Delegate | Mainnet Address | Testnet Address | Github link |
| --- | --- | --- | --- |
| **Aptos** | coming soon | **0x3e642b9d2845aa4efa3cdeb18acf6c7cd93112e4836a53e78f4b541e42a244b8** | coming soon |
| **Arbitrum** | coming soon | **0x270A49849E1400867a1343b4621c458d1F81190a** | coming soon |
| **Avalanche** | coming soon | **0x270A49849E1400867a1343b4621c458d1F81190a** | coming soon |
| **Base** | **0x5c2bfcf9c17CD53d55033769727196736CD188b3** | **0xD923a27A8b7cf8C9f3dFA022851B503a55b095a9** | coming soon |
| **BNB** | coming soon | **0x270A49849E1400867a1343b4621c458d1F81190a** | coming soon |
| **Ethereum** | coming soon | **0xc32d204404F33FF38Fee42394F7E671fd96314B3** | coming soon |
| **Fantom** | coming soon | **0x270A49849E1400867a1343b4621c458d1F81190a** | coming soon |
| **Linea** | coming soon | **0x270A49849E1400867a1343b4621c458d1F81190a** | coming soon |
| **Manta** | coming soon | **0x270A49849E1400867a1343b4621c458d1F81190a** | coming soon |
| **Mantle** | coming soon | **0x270A49849E1400867a1343b4621c458d1F81190a** | coming soon |
| **Optimism** | coming soon | **0x270A49849E1400867a1343b4621c458d1F81190a** | coming soon |
| **Polygon** | **0x3b3364656BbB7A23133e3f26D7F6850acfaAc394** | **0xd1673ef638084042A14e45d7e63e2a47e9d14E10** | coming soon |
| **Scroll** | coming soon | **0x270A49849E1400867a1343b4621c458d1F81190a** | coming soon |
| **Sei** | coming soon | **sei1dmwr4e6k4n0dlwtkh598sxp2al3wvkvwew658r3cqx98648uqhcs7sd38d** | coming soon |
| **Solana** | coming soon | **6rfCsexgBhW56iLQANo9MDV6dZGgdEnKLLVsrBwY3S1q** | coming soon |
| **Sui** | coming soon | coming soon | coming soon |
| **ZetaChain** | coming soon | **0xf8E1973814E66BF03002862C325305A5EeF98cc1** | coming soon |
| **zkSync** | coming soon | **0x4630e45Edb00298Eb7872FC5e237f6f0CE4995dF** | coming soon |

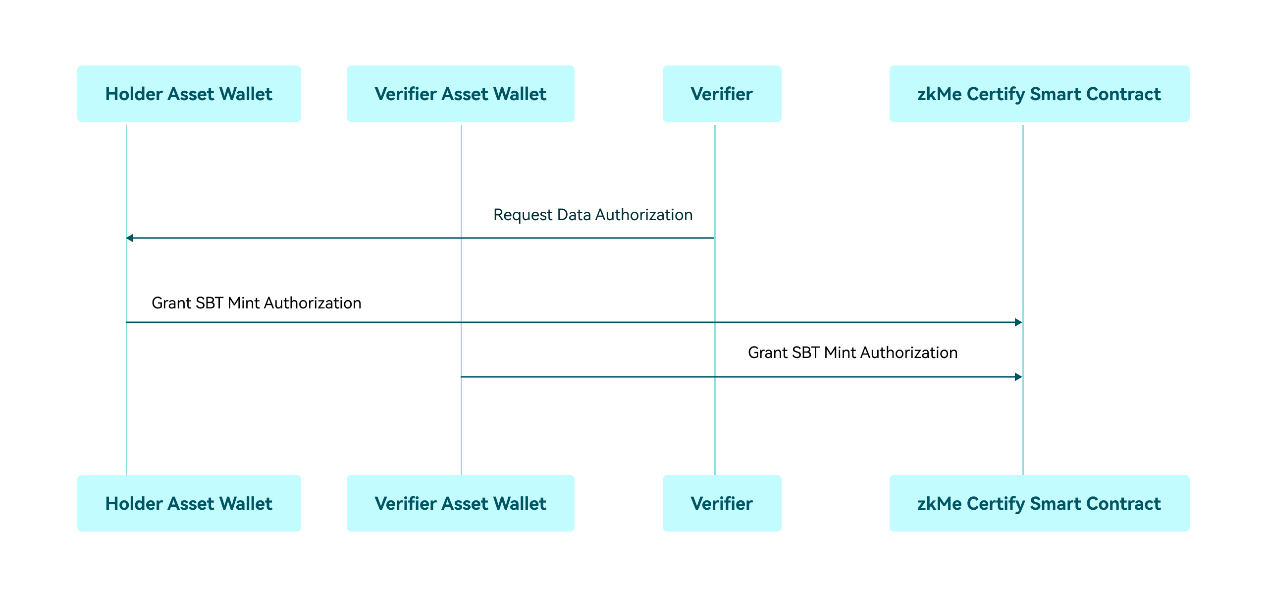
### zkMe Verify & Certify

The zkMe Verify & Certify Smart Contract (**ZKMEVerifyUpgradeable**) is multi-functional, capable of verifying users' eligibility for services and fulfilling data recovery requirements for compliance with regulators' rules. The SC is triggered once a dApp recognizes an SBT asset within a Holder's wallet.

The SC's Verify function is used for eligibility checks. It provides yes/no answers to a list of predetermined eligibility questions for each credential verified. These questions are outlined in the zkMe documentation on the zkMe website.



Following the verification, the SC's Certify function can be optionally triggered. It requires the explicit approval of the Holder (through transaction signature) and creates a verifier-specific copy of the Holder's SBT in a designated asset wallet. This copy includes the Holder's private key shard, enabling the Verifier to identify the Holder when a regulator initiates bad-actor proceedings, even without the Holder's approval.



Currently, the zkMe SC is compatible with the following chain ecosystems:

| zkMe Verify | Mainnet Address | Testnet Address | Github link |
| --- | --- | --- | --- |
| **Aptos** | coming soon | **0x3e642b9d2845aa4efa3cdeb18acf6c7cd93112e4836a53e78f4b541e42a244b8** | coming soon |
| **Arbitrum** | coming soon | **0xf8E1973814E66BF03002862C325305A5EeF98cc1** | coming soon |
| **Avalanche** | coming soon | **0xf8E1973814E66BF03002862C325305A5EeF98cc1** | coming soon |
| **Base** | **0x8c81bbc5cC9B6cdbb5c0e5DD8b9D5bfaF3575710** | **0xF58De9599C57bBAD68Fea0F39b73913daFcf0976** | coming soon |
| **BNB** | coming soon | **0xf8E1973814E66BF03002862C325305A5EeF98cc1** | coming soon |
| **Ethereum** | coming soon | **0xD231fF30102B34446035BA327ad4c596a5231cE3** | coming soon |
| **Fantom** | coming soon | **0xf8E1973814E66BF03002862C325305A5EeF98cc1** | coming soon |
| **Linea** | coming soon | **0xf8E1973814E66BF03002862C325305A5EeF98cc1** | coming soon |
| **Manta** | coming soon | **0xf8E1973814E66BF03002862C325305A5EeF98cc1** | coming soon |
| **Mantle** | coming soon | **0xf8E1973814E66BF03002862C325305A5EeF98cc1** | coming soon |
| **Optimism** | coming soon | **0xf8E1973814E66BF03002862C325305A5EeF98cc1** | coming soon |
| **Polygon** | **0x78D247ff4543Ef08488A1127034c2cE54B12A926** | **0xd1bB4Ac9a5864ebd521eCcCD8f459cfdF13e25b8** | coming soon |
| **Scroll** | coming soon | **0xf8E1973814E66BF03002862C325305A5EeF98cc1** | coming soon |
| **Sei** | coming soon | **sei1dmwr4e6k4n0dlwtkh598sxp2al3wvkvwew658r3cqx98648uqhcs7sd38d** | coming soon |
| **Solana** | coming soon | **6rfCsexgBhW56iLQANo9MDV6dZGgdEnKLLVsrBwY3S1q** | coming soon |
| **Sui** | coming soon | coming soon | coming soon |
| **ZetaChain** | coming soon | **0xA018F0593C1C3F62A68c3fc3B9D593961B207d96** | coming soon |
| **zkSync** | coming soon | **0xdbd9E736562584DB3fA1C5F39A47C2071DE0D5cb** | coming soon |

## zkMe Credential Schema

The credential schema on zkMe network is designed to be flexible and extensible, allowing issuers to customize the schema to meet their specific needs while maintaining compatibility with the W3C VC and VP data models; enabling Issuers to issue a wide range of credentials, from educational degrees to professional certifications, while ensuring that these credentials are interoperable and ease of use across different networks and applications.

zkMe specifies the following set of properties that its ZKPs (as VPs) must include as:

* **Issuer**: The entity that issued the verified.
* **Subject:** The DID to whom the credential pertains.
* **Type:** The type of credential being issued (e.g. Proof-of-Citizenship, Proof-of-Residence).
* **Issuance Date:** The date on which the credential was issued.
* **Expiration Date:** The date on which the credential expires.
* **Claims:** The specific eligibility the VP is attesting to (e.g. Adulthood - Is the holder over 18 years old?).
* **Proof:** A cryptographic proof that the credential was issued by the specified issuer and has not been tampered with since issuance.

In addition to these standard properties, VCs and VPs on zkMe may also include other properties or custom extensions, depending on the needs of the Issuer or the network's requirements as a whole. It's important to note that the content of VCs and VPs issued on zkMe is determined by the Issuer, and may vary depending on the type of credential being verified and the specific claims being attested.

In the zkMe zkKYC solution, the following information elements and descriptions are included:

**Issuer DIDs:** The public decentralized identifiers (DIDs) of the issuer are published in the Verifiable Data Registry. This information enables the holder to verify the authenticity and trustworthiness of the issuer.

**Holder DIDs:** The private DIDs of the Holder towards a particular Issuer are known only to the Holder and the Issuer. These DIDs enable the Holder to authenticate themselves to the Issuer and provide proof of their Identity.