Comparative analysis of EVM compatible DID methods

note: the DID methods were extracted from this list: https://www.w3.org/TR/did-extensions-methods/

List of methods supported by DIF listed under "Drivers"

note: requirements are extracted from felix's input but maybe wrongly understood by me

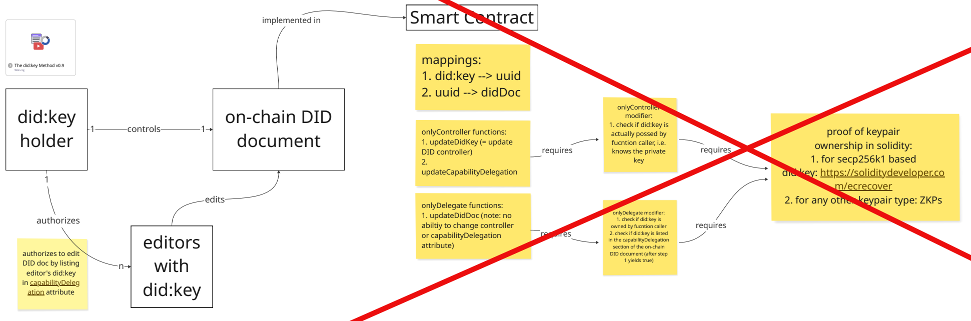
DID method	Info: usable (e.g. supported by Altme, project still active)?	Info: part of DIF universal resolver	Requriement: updateable did document	Requriement: multiple DID controllers --> key rotation thus possible	Requirement: multiple keys possible in verificationMethod and authentication section	Soft requirement: if DID method is not compatabile with altme, linkage of DID to did:key must be possible (ideally with proof of ownership)
zkMe		not listed				
iden3		compliant				
ev	unknown	no response				
kaname	unknown	not listed				
polygonid (same as iden3???)	supported by altme		Yes	Yes	Yes	
ethr	active, supported	Yes	Yes, with delegates and attributes	Yes, changeOwner, delegates	Yes, unlimited	compatabile with altme so no need for did:key

From Altme Repo: "Talao DID resolver for did:web, did:ethr, did:ebsi, did:dht, did:sov, did:cheqd...,"

multisig is not in the DID document specification itself. However functionally, that is fully supported through Ethereum native multisig contracts. This is actually even more flexible than built in multisig.

did:ethr or is that missing on purpose? -felix

~~Solution 1: did:key with on-chain DID documents through custom smart contract~~



~~authorizes to edit DID doc by listing editor's did:key in capabilityDelegation attribute~~

~~mappings:~~
~~1. did:key --> uuid~~
~~2. uuid --> didDoc~~

~~onlyController functions:~~
~~1. updateDidKey (= update DID controller)~~
~~2. updateCapabilityDelegation~~

~~onlyDelegate functions:~~
~~1. updateDidDoc (note: no abiltiy to change controller or capabilityDelegation attribute)~~

~~onlyController modifier:~~
~~1. check if did:key is actually possed by fucntion caller, i.e. knows the private key~~

~~onlyDelegate modifier:~~
~~1. check if did:key is owned by fucntion caller~~
~~2. check if did:key is listed in the capabilityDelegation section of the on-chain DID document (after step 1 yields true)~~

~~proof of keypair ownership in solidity:~~
~~1. for secp256k1 based did:key: https://soliditydeveloper.com/ecrecover~~
~~2. for any other keypair type: ZKPs~~

~~Pro~~

- ~~custom solution --> independence from external solutions~~
- ~~potential for funding because of enhancing did:key method~~
- ~~did:key is most interoperable DID method~~
- ~~ZKP related expertise is gained that possibly helps solving problem 2 of this project: on-chain VC veriifcation with privacy~~

~~Contra~~

- ~~no funding for did method~~
- ~~complexity --> maybe no time for the possibly more important on-chain VC verification (problem two of this project)~~

Solution 2: Existing Standard: did:ethr (ERC-1056 Implementation)

Proposal: EVM-Compatible DID Method (did:ethr)

Decision: Use the did:ethr method implementation based on the canonical ERC-1056 Smart Contract standard deployed on Tezos Etherlink.
https://eips.ethereum.org/EIPS/eip-1056
https://github.com/uport-project/ethr-did-registry

Why did:ethr fits our Hard Requirements:

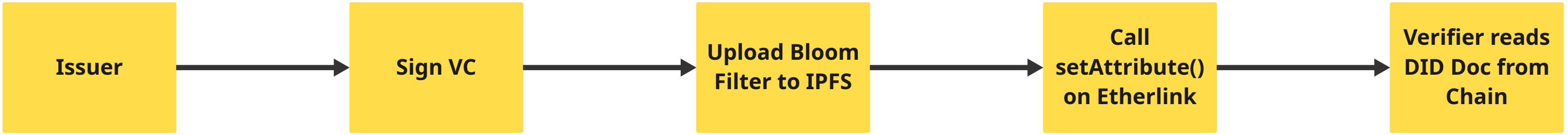
1. On-chain and EVM-based
- did:ethr is purely smart-contract based.
- It does not require a side-chain or specialized nodes; it runs directly on Etherlink L2.

2. Supports key management & services
Services: The contract has a native setAttribute function to add service endpoints (e.g., CRSetRegistry pointing to IPFS).
Keys: Supports multiple keys for different purposes (verification, authentication).

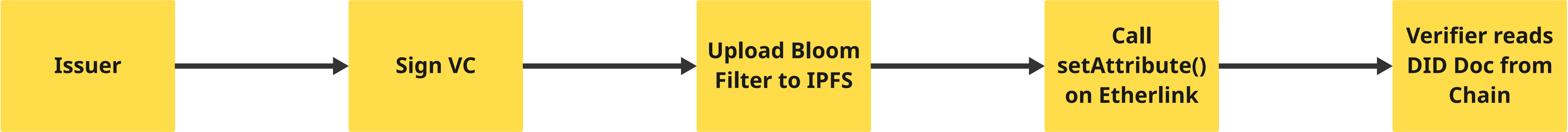
3. Revocation is a service entry
- We can update the DID Document to include a service of type RevocationRegistry containing the IPFS CID of the latest Bloom Filter.
- This update is a simple transaction to the Etherlink contract.

4. No static key that owns the DID
Key Rotation: The identityOwner (controller) can be changed anytime via changeOwner.
Safety: The DID identifier (address) remains permanent, even if the controlling private key is rotated (e.g., from a compromised key to a secure Ledger).

5. Delegation
- ERC-1056 separates **Identity Owner** (Management) from **Delegates** (Signing).
- *Example:* A cold wallet holds ownership, while a server-side hot wallet is added as a delegate just for signing VCs (valid for X seconds).



Issuer
Sign VC
Upload Bloom Filter to IPFS
Call setAttribute() on Etherlink
Verifier reads DID Doc from Chain


Issuer
Sign VC
Upload Bloom Filter to IPFS
Call setAttribute() on Etherlink
Verifier reads DID Doc from Chain