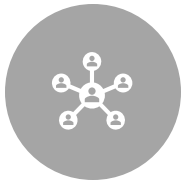
rubix

RUBIX DID – Own Your Identity

Self-Signed Identity

Existence — Users have independent existence.



Control — Users control their identities.

Access — Users have access to their own data & control who else can access them

Portability — Information and services about identity are transportable

Consent — Users must agree to the use of their identity.

Minimization — Users can disclose only selected information

rubix

Register self-owned identities

Create

- Create Unique IDentity (UID) and derive shares using Non-Linear Secret Sharing (NLSS)

Map

- Map any digital information to UID

Prove

- Prove ownership of UID by proving ownership of secret share (ZK proofs)

rubix

Verifiable Claims

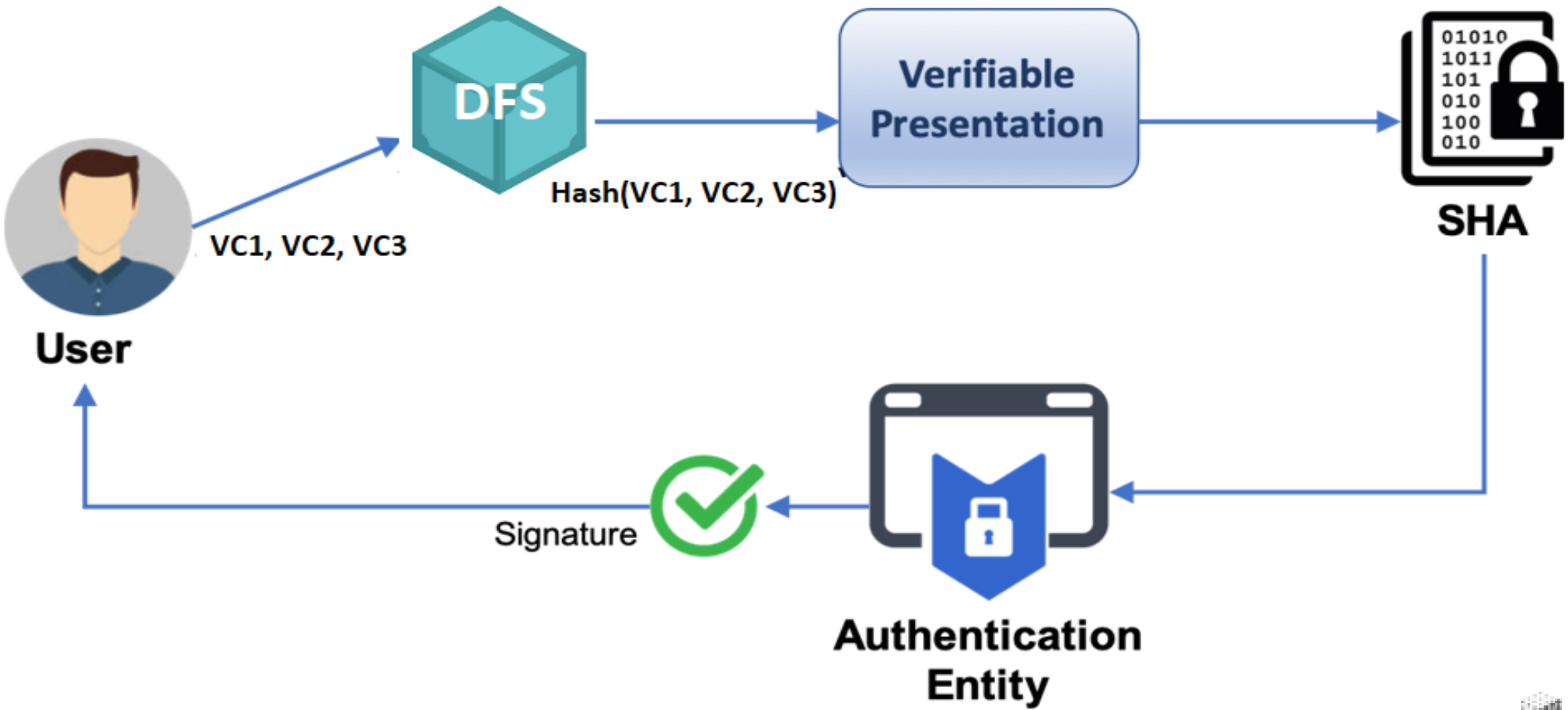
GET SIGNED BY VERIFIERS BY SUBMITTING VERIFIABLE PRESENTATIONS

CAN GET MULTIPLE VERIFICATIONS FOR SAME VERIFIABLE CREDENTIALS

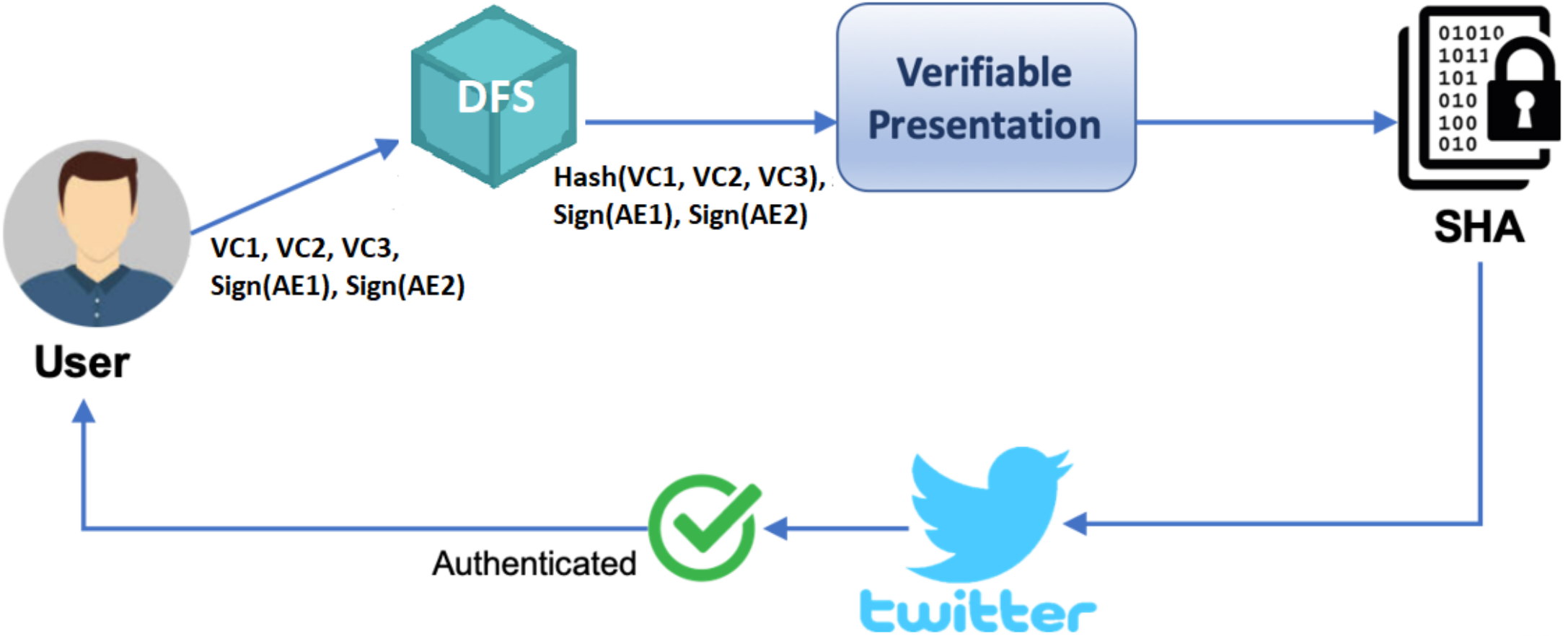
USES NLSS BASED CHALLENGE – RESPONSE TO ACHIEVE SELECTIVE DISCLOSURE

VERIFIABLE CREDENTIALS IN JSON/XML FORM

rubix


DFS
Verifiable Presentation
01010
1011
101
010
100
010
SHA
User
VC1, VC2, VC3
Hash(VC1, VC2, VC3)
Signature
Authentication Entity

rubix


DFS
Verifiable Presentation
01010
1011
101
010
100
010
SHA
User
VC1, VC2, VC3,
Sign(AE1), Sign(AE2)
Hash(VC1, VC2, VC3),
Sign(AE1), Sign(AE2)
Authenticated
twitter

rubix

Privacy and Security Features

ULTRA-SCALABLE - < 100MS FOR VERIFICATION (MILLIONS OF ASYNCHRONOUSLY PARALLEL VERIFICATIONS)

PRIVACY & FAIRNESS – SELECTIVE DISCLOSURE WITH TAMPER-PROOF, IRREFUTABLE SIGNATURES

KEY RECOVERY BY NLSS SCHEMA

KEY DERIVATION & RECOVERY USING BIOMETRICS FOR ADDED SECURITY

rubix

RUBIX DID vs Existing models (Everynym, Blockcerts, Microsoft ION…)

RUBIX DID

- Complete layer-1 solution (on-chain scaling)
- Content based addressing (integrity over credentials)
- Instant finality and confirmation
- Independent verification of each transaction
- Millions of parallel updates per second

EXISITING MODELS

- Layer-2 solution built on Bitcoin and/or Ethereum
- Location based addressing (integrity over location of credentials)
- Confirmation dependent on when the transaction is added by layer 1 network
- Transactions are pooled to reduce cost
- Constrained by parent chain scaling

rubix

Crucial Security Flaws in Current Models

DID solutions like blockcerts create certificates in json format and push the json file to layer 1 blockchain

The json file sometimes holds URL's of data/credentials of the certificates as value. For example a certificate could be

{
Date issued: 29 Jan 2020
Name : John Doe
Details : www.someurl.com/files/json/125442
}

This data hashed and pushed into blockchain , only protects the integrity of URL www.someurl.com/files/json/125442 . The data inside the URL can be modified, hence does not stand true to its immutability claims

rubix

RubiX DID – Data Integrity With On-Chain Scaling

Rubix DID uses Real world DFS based on Content-Based Addressing(CBA), preserves complete integrity over signed data.

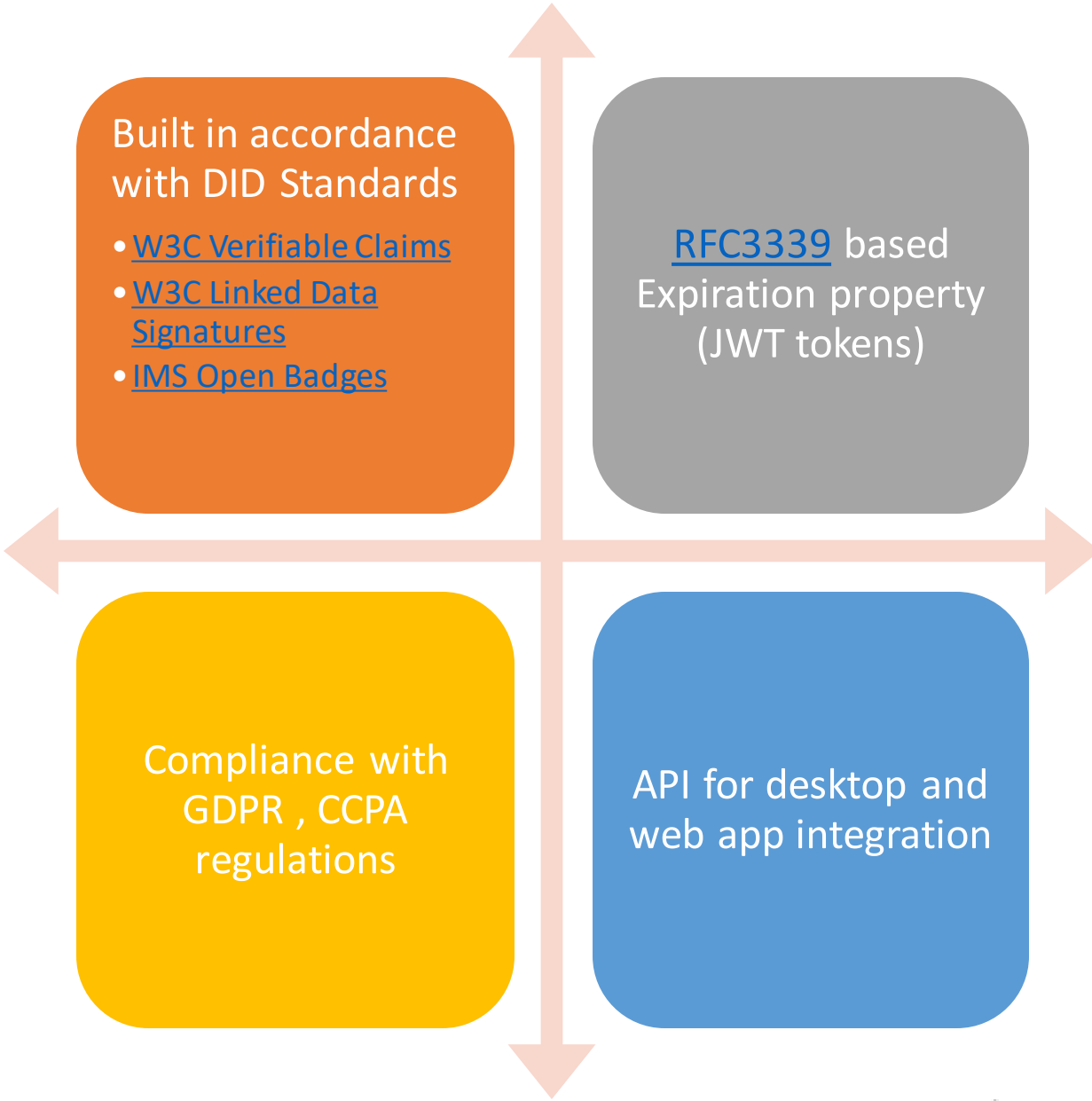
Rubix DID credentials build on json structure uses CBA databases to hold integrity over data instead of the location of data. For instance,

{
Date : 29 Jan 2020
Name : John Doe
Details : QmVBdbYa6GXXtG2JVK2NDkNKsbLeWzDPUdpsLoA5yYvJHu
}

The multihash QmVBdbYa6GXXtG2JVK2NDkNKsbLeWzDPUdpsLoA5yYvJHu is the hash of the content, created using distributed CBA database, any change in data will result in hash mismatch.

rubix

Interoperability and Open Standards



rubix

THANK YOU

rubix

rubix.network

Info@rubix.network