

7th INTERNATIONAL CONFERENCE ON PUBLIC KEY INFRASTRUCTURE AND ITS APPLICATIONS (PKIA 2026)

SEPTEMBER 10-11th, 2026

PKI-Anchored Self-Sovereign Identity for Certificate Issuance & Verification Using DSC Hardware Wallets on Polygon Blockchain

SHRAWAN KUMAR, RAS DWIVEDI, ANUSHKA PANDEY
IHUB NTIHAC FOUNDATION, IIT KANPUR

MOTIVATION

Two paradigms for digital identity -- each with a gap

Digital Signature Certificates

- Strong identity assurance via CA-verified, government-issued credentials
- Root-CA compromise forces reissuance across the entire trust hierarchy
- Certificate validity at issuance never guarantees the holder is still the rightful entity
- Centralized, limited interoperability with decentralized identity systems

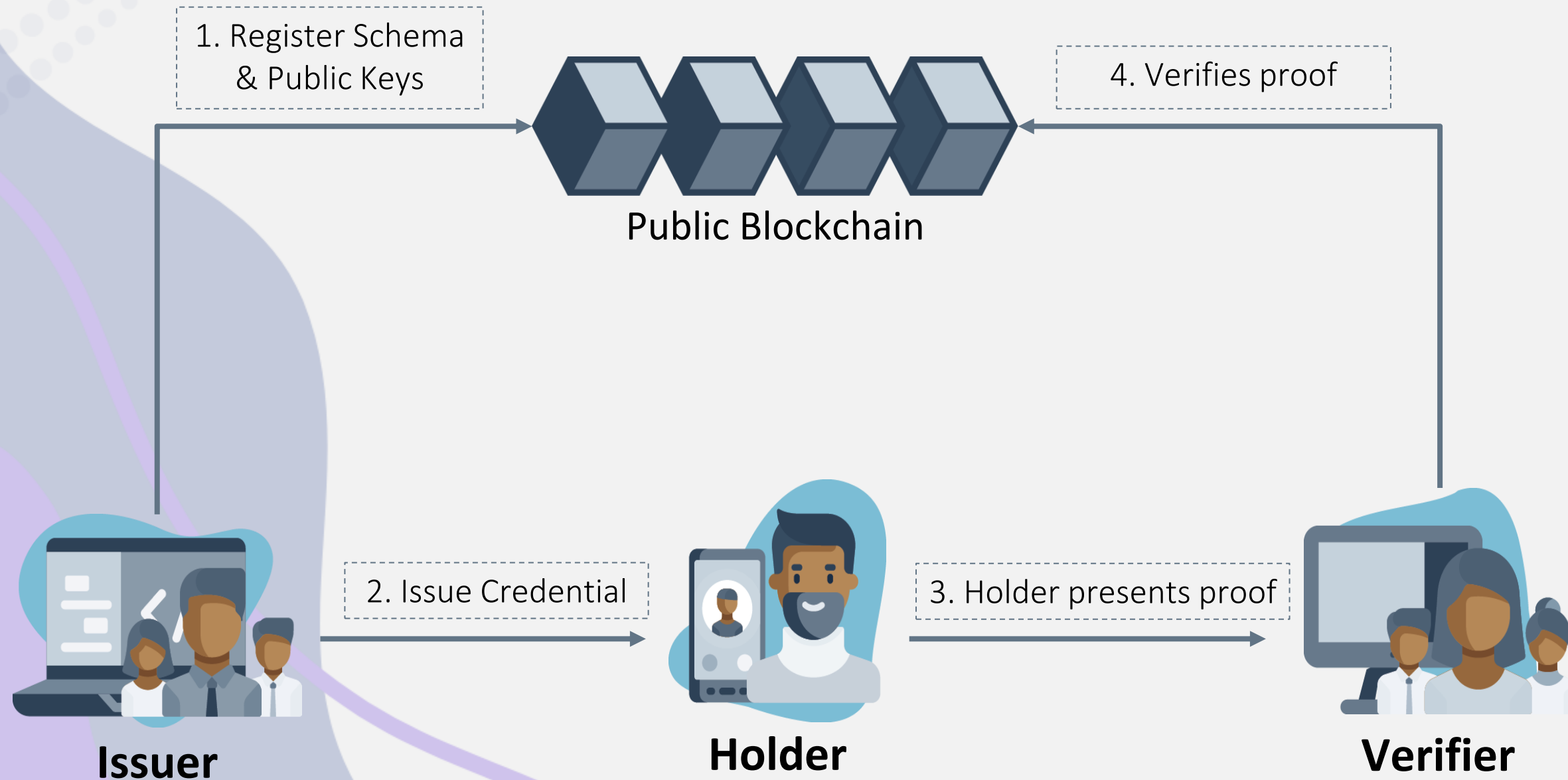
Self-Sovereign Identity

- Decentralized, holder-controlled credentials with strong interoperability
- Most implementations use self-generated keys in software wallets (e.g. MetaMask)
- No binding to a verified real-world identity at key generation
- No hardware-level protection of the signing key

Neither paradigm alone delivers credentials that are simultaneously high-assurance, hardware-protected, and fully decentralized.

BACKGROUND

The SSI trust model, and the DSC hardware root of trust



The DSC Hardware Token

- Issued by a CCA-licensed CA after in-person / video KYC of the subscriber
- Key pair generated on-device; the private key never leaves the hardware
- Conforms to PKCS#11 and FIPS 140-2 Level 2 (or higher)
- Signing happens inside the token — host passes data, receives signature only
- Valid for 1–3 years, after which a fresh key pair is issued on renewal

Associated Terms:

- DID
- Verifiable Credentials
- Verifiable Presentations
- Issuer/Holder/Verifier
- Trust Registry (Blockchain)

Note: The issued credential is stored on the mobile device of the holder

OUR APPROACH

Repurpose DSC hardware tokens as PKI-anchored SSI wallets

A DSC hardware token provides the identity assurance and hardware key protection; Polygon provides decentralized, runtime-CA-free verification.



did:dsc DID method

A persistent, holder-controlled DID whose identifier survives DSC renewal — only the mapped public key changes.



Smart-contract-enforced key rotation

Rotation is authorized solely by a proof-of-possession signature; the backend relay has no authority over identity.



PKI Binding Proof

A cryptographic link between a DID and a CA-certified DSC public key, without on-chain X.509 verification.



Runtime CA-independent verification

DID resolution, signature checks, and revocation are performed locally or on-chain — no CA contact at verify time.



Hardware-backed SSI wallet

Repurposes India's existing PKCS#11 DSC token infrastructure — no new hardware wallet required.

THE DID:DSC DID METHOD

A persistent identifier, decoupled from the signing key

did:dsc:`<network>:<id>`

`id = keccak256( PK_DSC, initial )`

Stable identifier

The DID string is derived once, from the initial DSC public key, and never changes.

Evolvable key mapping

The current public key is looked up on-chain — key rotation never changes the DID string.

Zero-gas resolution

resolveDID is a read-only view call, returning the live key plus the PKI Binding Proof.

Because credentials reference the DID string — not the key — a credential issued to a holder's did:dsc survives every future DSC renewal.

Role-based keys: why issuers use did:ethr, not did:dsc

- Issuers must **write** to Polygon -- publish schemas, update DID documents, write revocation entries -- which needs a native secp256k1 Ethereum key, not just a proof-of-possession signature.
- Holders only ever **prove key possession** to register or rotate a did:dsc identifier, so a relay can submit that narrow transaction for them -- no Ethereum key needed.
- The issuer's did:ethr DID still carries a **PKI Binding Proof** -- the same construction used for holders -- so it inherits full CA-grade identity assurance despite using an Ethereum-native key.

Registration once -- self-sovereign rotation on every renewal

REGISTRATION · Eq. 5–7

```

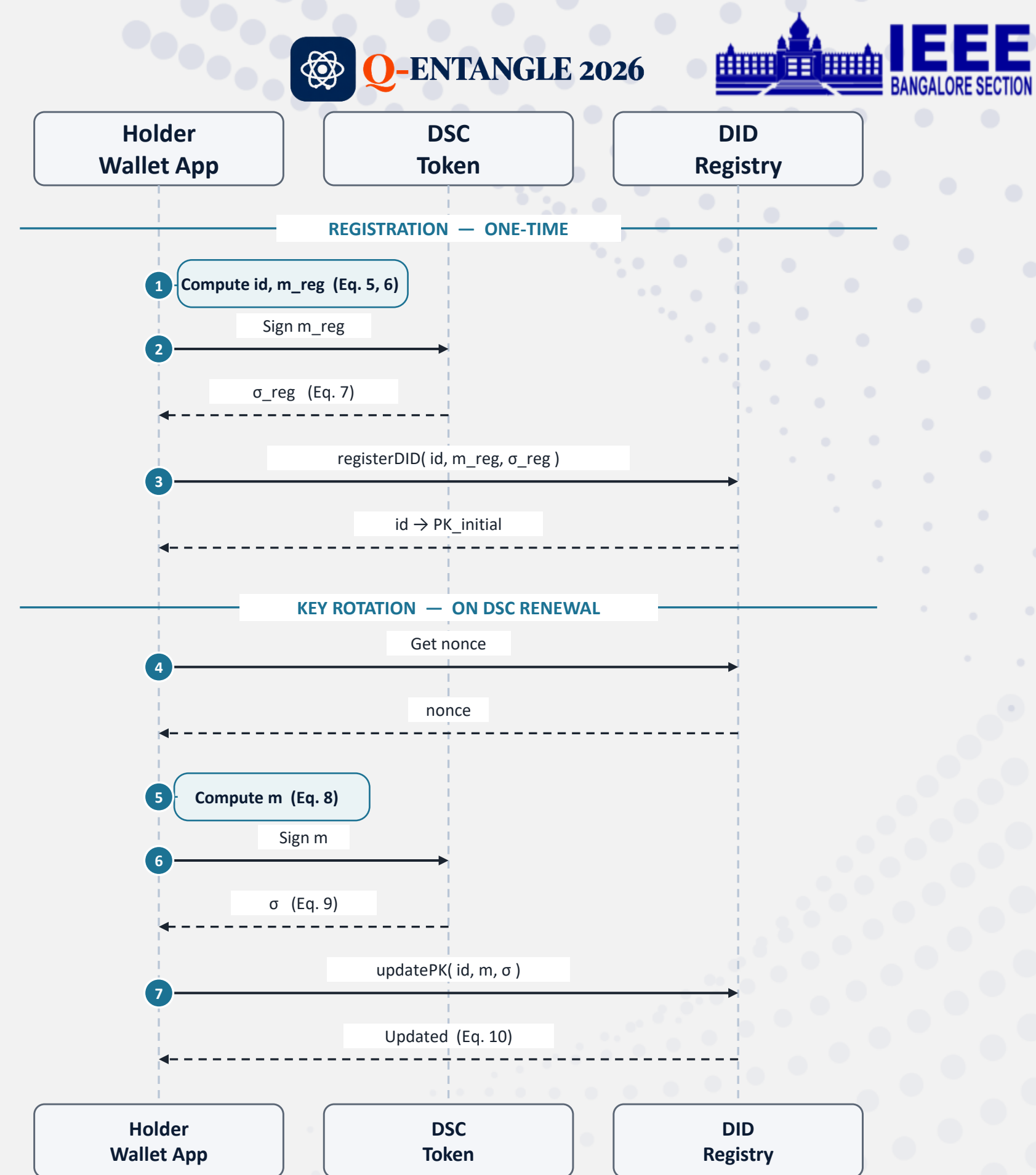
id = keccak256( PK_DSC, initial )
m_reg = keccak256( id || PK_DSC, initial )
σ_reg = Sign( SK_DSC , m_reg )
    
```

KEY ROTATION · Eq. 8–10

```

m = keccak256( id || PK_new || nonce )
σ = Sign( SK_DSC, current , m )
didToCurrentPK[id] = PK_new ; nonce ← nonce+1
    
```

Same DID, new key: the contract verifies the rotation signature against the currently registered key, updates the mapping, and increments a nonce — the relay that submits the transaction contributes no authority over the outcome.



CRYPTOGRAPHIC FOUNDATION

The PKI Binding Proof: CA trust without on-chain X.509

Construction $\Pi = (\sigma_{\text{bind}}, h_{\text{cert}}, \text{uriC})$

m_bind = SHA-256(DID)

binding message over the DID string

σ_{bind} = Sign(SK_DSC , m_bind)

signed inside the DSC hardware token

h_cert = keccak256(C)

commitment to the full DER certificate chain

Chain C stored off-chain ($\approx 3-5$ KB via content-addressed URI); only h_cert is committed on-chain.

Verification (by any verifier, offline-capable)

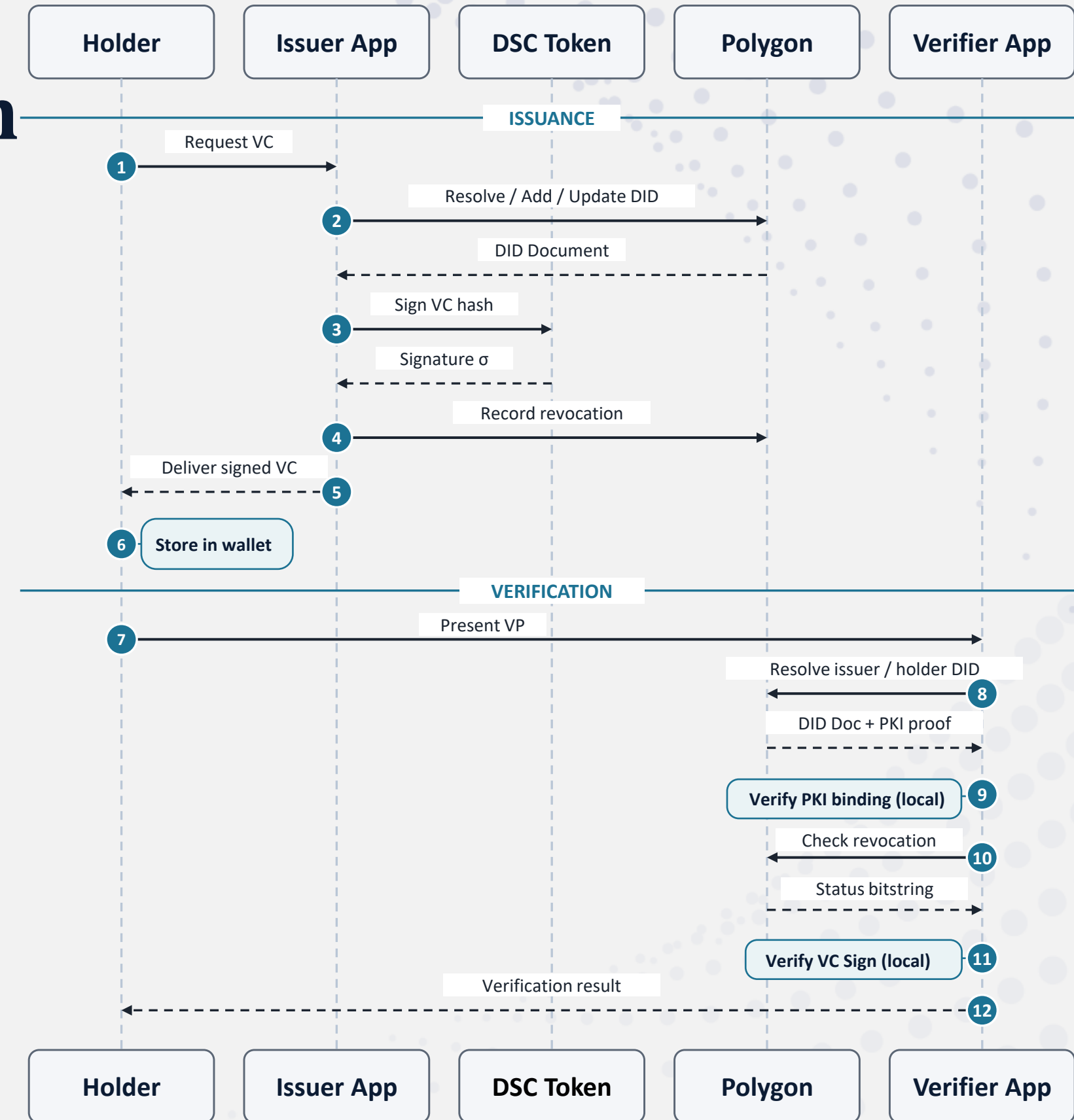
- Retrieve C from uriC and confirm $\text{keccak256}(C) = h_{\text{cert}}$
- Validate C to a locally trusted root CA (the CCA root)
- Extract PK_DSC from the leaf certificate
- Verify the binding signature over m_bind against PK_DSC

A successful check proves the DID controller also held the CA-certified private key — inheriting the CA's identity assurance for the DID.

END-TO-END FLOW

Credential Issuance and Verification

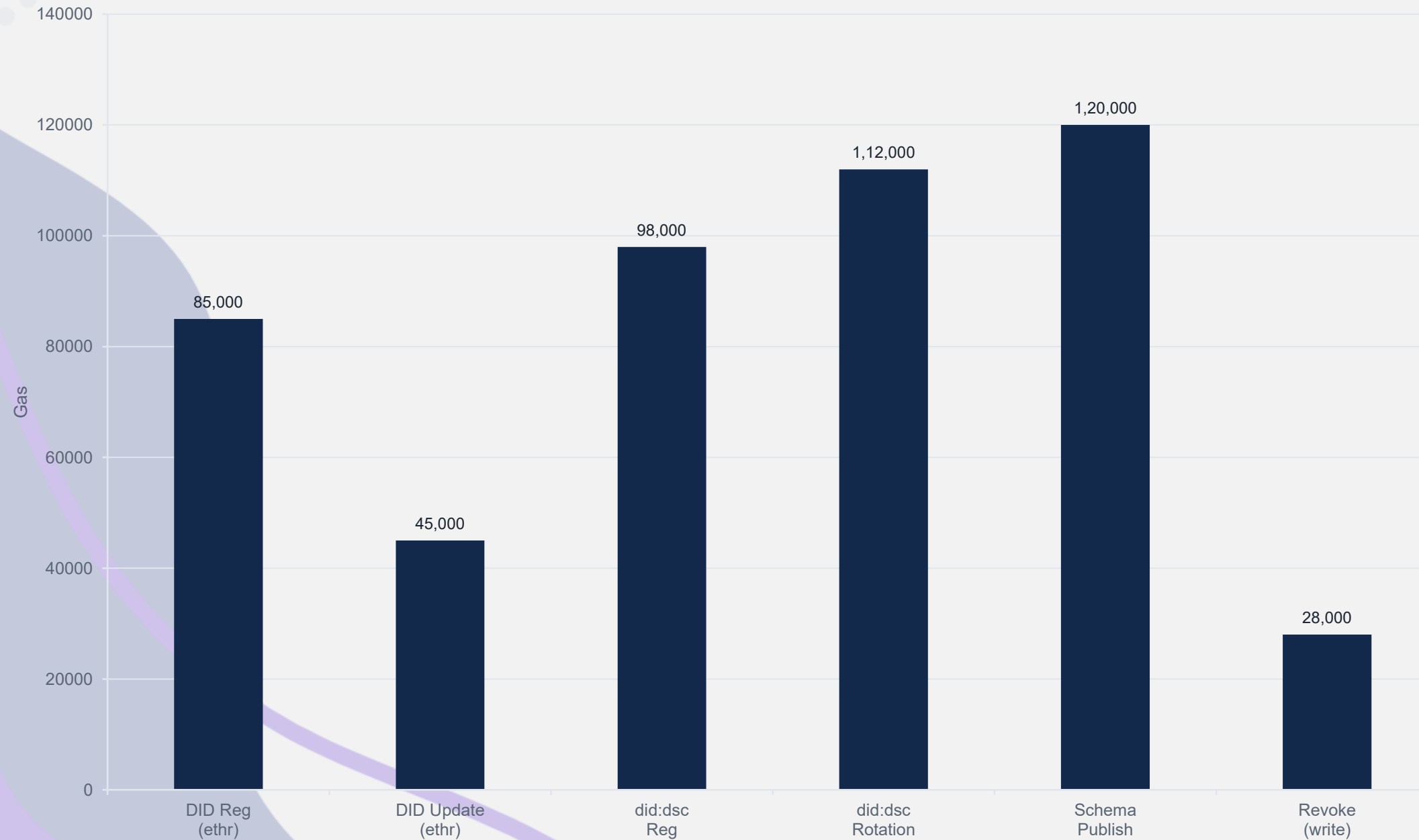
No third-party contact at verification: PKI-binding validation and VC signature verification are both performed locally by the verifier.



EXPERIMENTAL RESULTS

On-chain transaction costs are low across every operation

Gas Cost by Operation



Step	Min (ms)	Avg (ms)	Max (ms)
VP Parsing	8	12	21
DID Resolution	190	295	460
Signature Verification	22	28	45
Revocation Check	110	160	290
PKI Chain Validation	85	115	200
Total (E2E)	415	610	1016

DID resolution & revocation checks are read-only view calls, so they cost zero gas; the highest-cost operation is Schema Publication at ~120,000 gas — a one-time cost per credential type, not per credential issued.

EXPERIMENTAL RESULTS

How the proposed approach compares

Approach	ID Assurance	HW Key	Decentralization	Runtime CA	Cost
PKI Only	High	Yes	No	Required	Med
SSI	None	No	Yes	No	Low
Proposed (DSC + SSI)	High	Yes	Yes	No	Low

The proposed DSC + SSI approach is the only one achieving high identity assurance, hardware key protection, full decentralization, no runtime CA dependency, and low cost — simultaneously.

SECURITY ANALYSIS

Five guarantees, verified independently of the backend relay



Key security

Private keys never leave the hardware token.
Compromising the host machine cannot extract the signing key.



Rotation integrity

Only the current DSC key holder can update the DID mapping. The relay can delay a rotation but never forge one.



Identity binding

A did:dsc DID is seeded from the CA-certified key at genesis; rotations are authorized only by the prior key holder.



Revocation security

Revocation state lives on-chain, writable only by the issuer's DID controller — no reliance on OCSP responders.



Replay prevention

Rotation messages carry a monotonically increasing nonce; VPs bind to a verifier-supplied challenge.

CONCLUSION

did:dsc establishes a practical path to high-assurance SSI

Government-issued DSC hardware tokens, repurposed as SSI wallets and anchored on Polygon, deliver identity assurance inherited from PKI, hardware-level key protection, persistent DIDs across the DSC lifecycle, and fully decentralized runtime verification — at the cost of a standard government DSC token.

FUTURE WORK

Post-quantum readiness

Pluggable verification supports CRYSTALS-Dilithium, Falcon, or SPHINCS+ once DSC issuance adopts them.

Multi-issuer composition

Combining credentials from multiple issuers into a single verifiable presentation.

Formal verification

Formal verification of the did:dsc smart contract's proof-of-possession logic.

THANK YOU