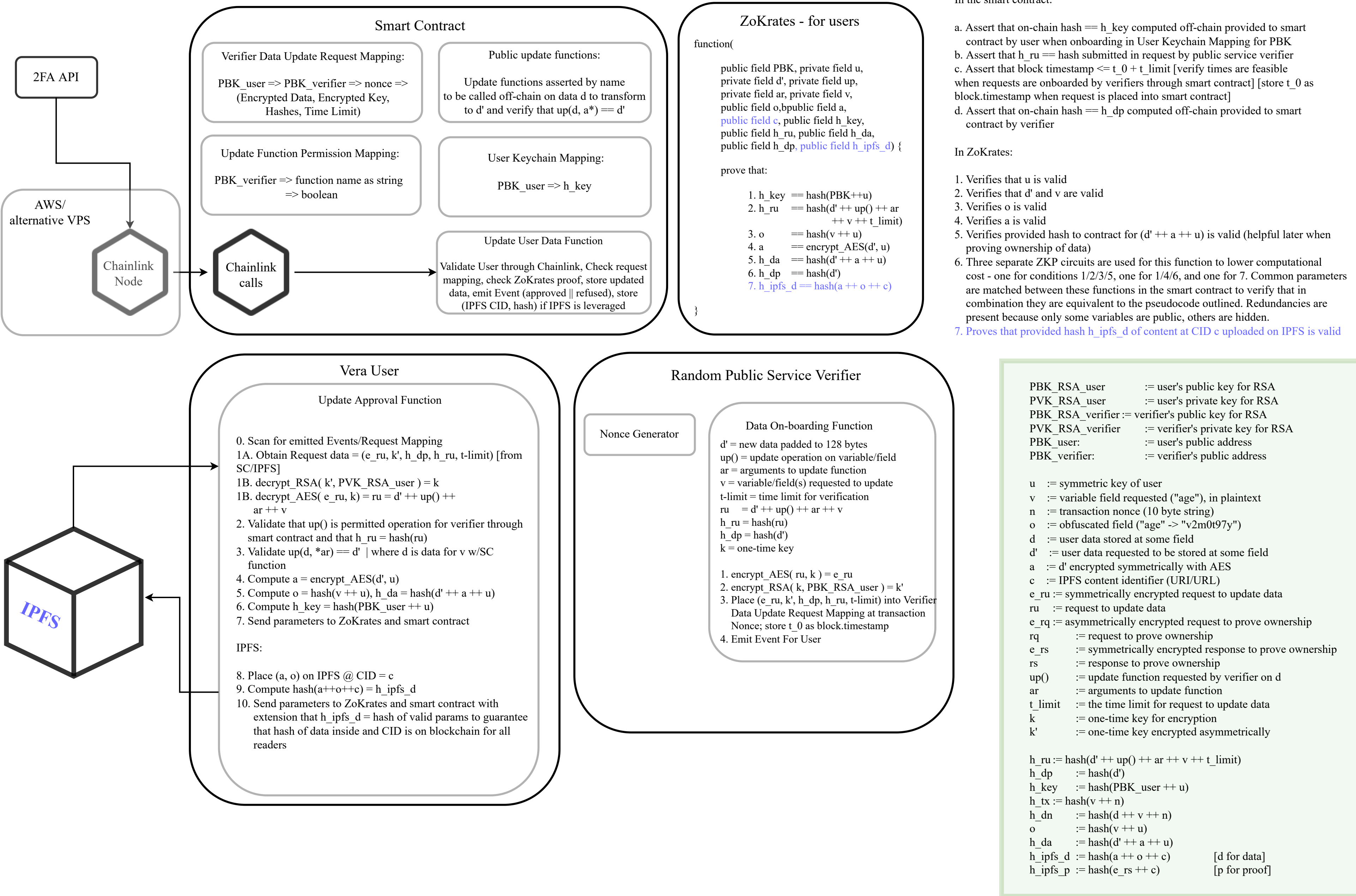
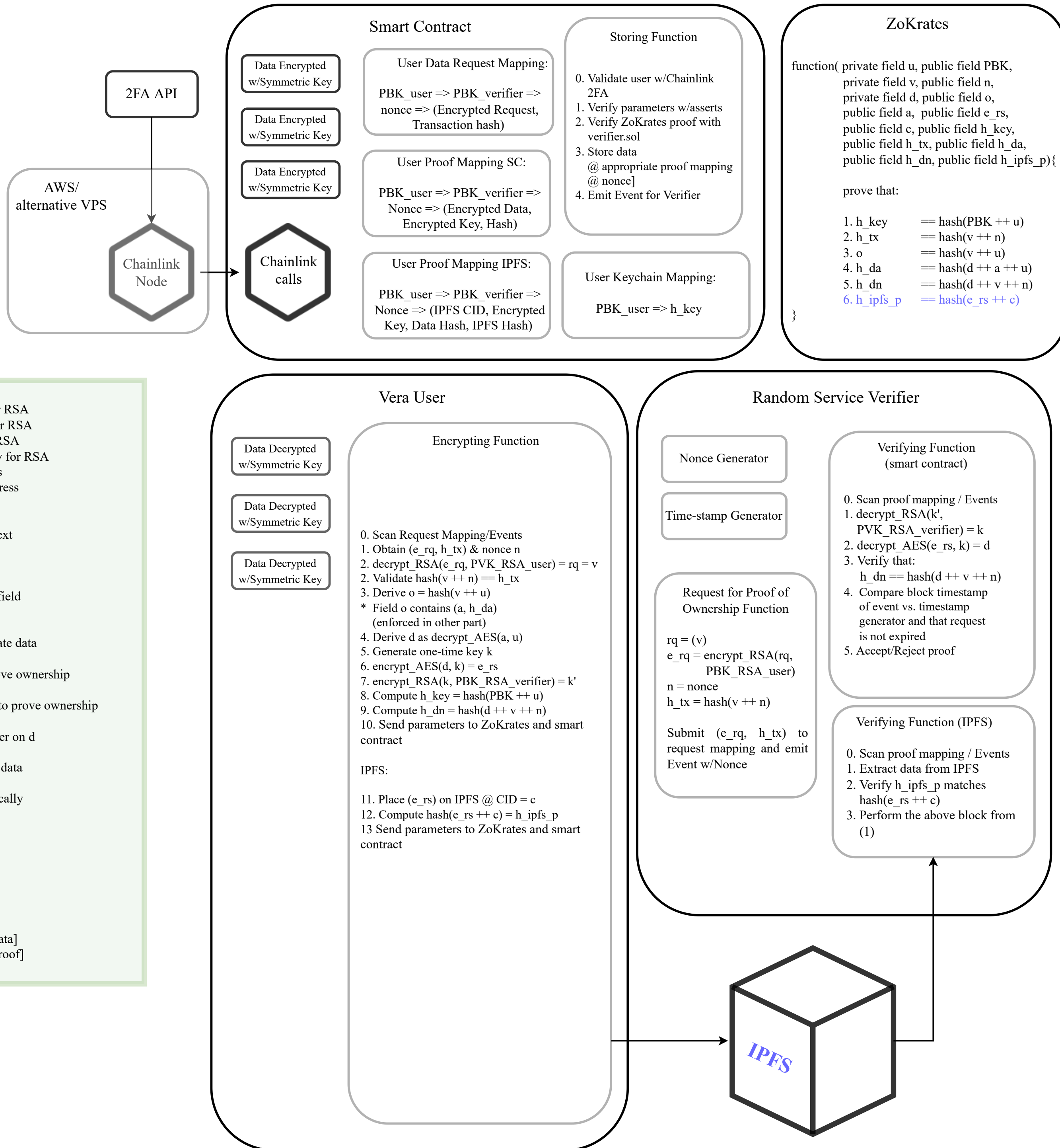


Secure On-boarding of Data To Smart Contract While Maintaining Privacy in Solidity



Proving Ownership of Encrypted Data In Smart Contract While Maintaining Privacy in Solidity



In the smart contract:

- a. Assert that on-chain hash == h_key computed off-chain provided to smart contract by user when onboarding in User Keychain Mapping for PBK
- b. Assert that on-chain hash == h_tx computed off-chain provided to smart contract by public service verifier
- c. Assert that User Data Mapping at (PBK_user) => o => (a, h_da)

In ZoKrates:

1. Therefore the symmetric key u provided is valid
2. Therefore v (and n) is valid and request is decrypted correctly [coupled with assertion (b)]
3. Therefore o is valid, the obfuscated field from which we are retrieving data
4. Therefore (d, a) is valid [one MUST onboard with valid hash h_da]
5. Therefore h_dn is a valid hash of d with v and n that can be used to verify that the encrypted response e_rs is valid upon decryption to rs
6. Proves that provided hash h_ipfs_p of content at CID c uploaded on IPFS is valid

In the smart contract:

1. Place (e_rs, k', h_dn) into smart contract w/storing function
2. Place (IPFS hash [c], k', h_dn, h_ipfs_p) into smart contract w/storing function

Off-chain:

1. a. Extract (e_rs, k', h_dn, h_ipfs_p) from smart contract/IPFS CID
b. Verify $\text{hash}(e_rs ++ c) == h_ipfs_p$
2. decrypt_RSA(k', PVK_RSA_verifier) = k
3. a. decrypt_AES(e_rs, k) = d
b. Verify that $h_dn == \text{hash}(d ++ v ++ n)$
c. Compare timestamps of proof
4. Decide to accept or reject proof

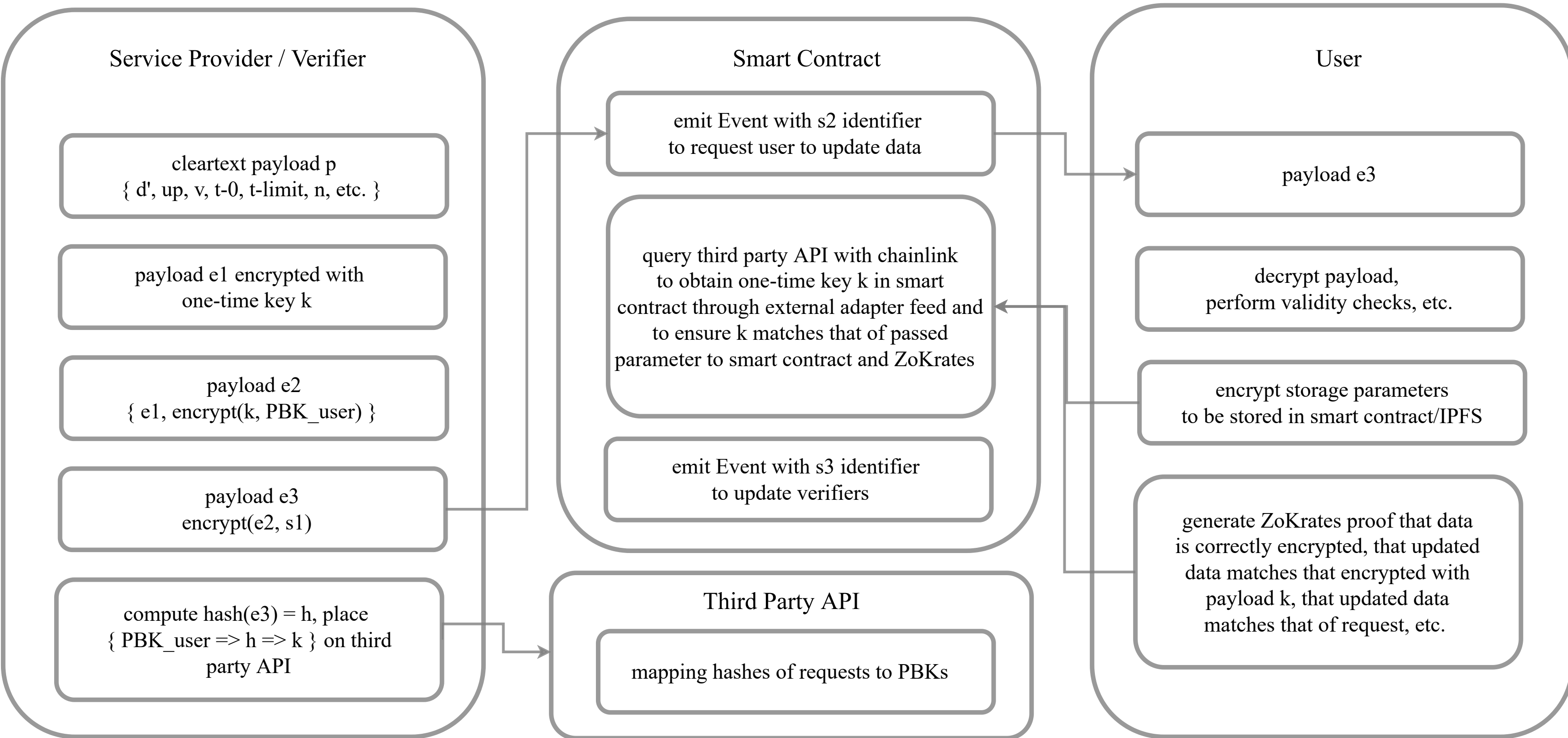
PBK_RSA_user := user's public key for RSA
PVK_RSA_user := user's private key for RSA
PBK_RSA_verifier := verifier's public key for RSA
PVK_RSA_verifier := verifier's private key for RSA
PBK_user: := user's public address
PBK_verifier: := verifier's public address

u := symmetric key of user
v := variable field requested ("age"), in plaintext
n := transaction nonce (10 byte string)
o := obfuscated field ("age" -> "v2m0t97y")
d := user data stored at some field
d' := user data requested to be stored at some field
a := d' encrypted symmetrically with AES
c := IPFS content identifier (URI/URL)
e_ru := symmetrically encrypted request to update data
ru := request to update data
e_rq := asymmetrically encrypted request to prove ownership
rq := request to prove ownership
e_rs := symmetrically encrypted response to prove ownership
rs := response to prove ownership
up() := update function requested by verifier on d
ar := arguments to update function
t_limit := the time limit for request to update data
k := one-time key for encryption
k' := one-time key encrypted asymmetrically

h_ru := hash(d' ++ up() ++ ar ++ v ++ t_limit)
h_dp := hash(d')
h_key := hash(PBK_user ++ u)
h_tx := hash(v ++ n)
h_dn := hash(d ++ v ++ n)
o := hash(v ++ u)
h_da := hash(d' ++ a ++ u)
h_ipfs_d := hash(a ++ o ++ c) [d for data]
h_ipfs_p := hash(e_rs ++ c) [p for proof]

Accounting For Side-Channel Inference: An Implementable Extension With Chainlink and/or Asymmetric Encryption Verification Components To Hide Vera Request and Response Receivers For Proof of Valid Storage When On-boarding

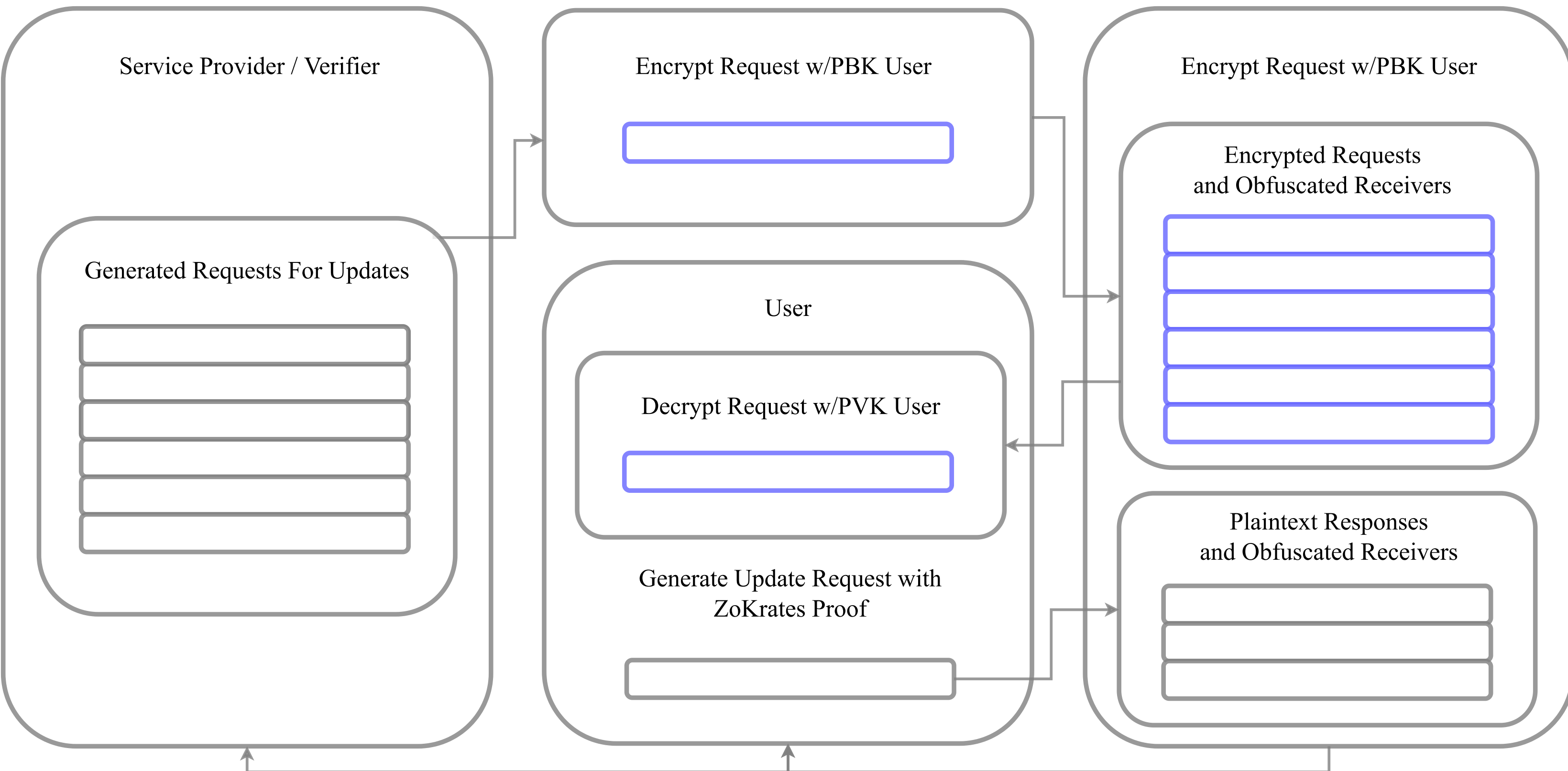
**The requests, responses, and fields remain obfuscated regardless of whether this extension is present,*



Assume a number of negotiated shared secrets { s1, s2, s3, ... } between user and verifier are available negotiated through DH key exchange, secondary secure channels, etc.

Accounting For Side-Channel Inference: An Implementable Extension With Diffie-Hellman Exchange And/Or
Secondary Secure Channel To Hide Vera Request and Response Receivers For Proof of Ownership

**The requests, responses, and fields remain obfuscated regardless of whether this extension is present,*



Alert Each Other Of Available Request
Through Secure Channel Or Through
Emitting Events/Information With Shared
Secret In Smart Contract Such as With
SmartDHX Diffie-Hellman Key Exchange
(Using A Separately Negotiated DH Key for
Each Direction), etc.

Alternatively, The User/Verifier
Periodically Monitors The Blockchain
To Identify Matching Requests