CMTAT-Confidential

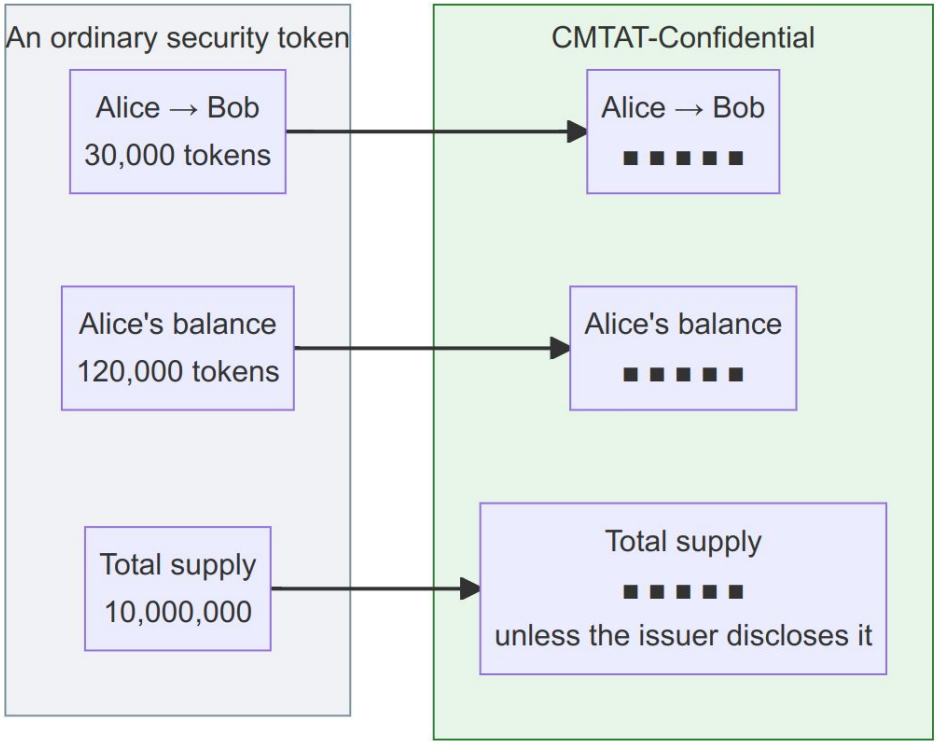
From public amounts to confidential ones

Ryan
(Taurus/CMTA)

What you will learn today

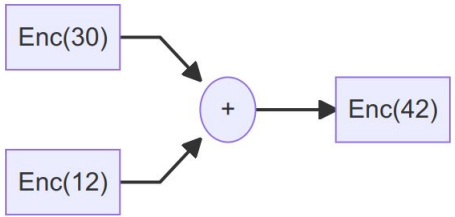
- What CMTAT-Confidential is for
- How the technology works: FHE, the coprocessors, and who holds the keys
- What the token is: CMTAT + ERC-7984, what is encrypted and what is not
- Who can read what: permissions, observers, the total supply

Public to Confidential



How the technology works

How Zama works — computing without decrypting



- Fully Homomorphic Encryption (FHE) computes directly on encrypted data, called a ciphertext, without ever decrypting it.

  Example: Enc(a) + Enc(b) = Enc(a + b)

- Data is encrypted with a global public key, the same one for the whole network,

  - Any contract can compute on any other contract's encrypted values (composability).

How Zama works: computation

- The computations are run by the coprocessors (5 operators)

    - They work with the evaluation key, a public key that computes but cannot decrypt.

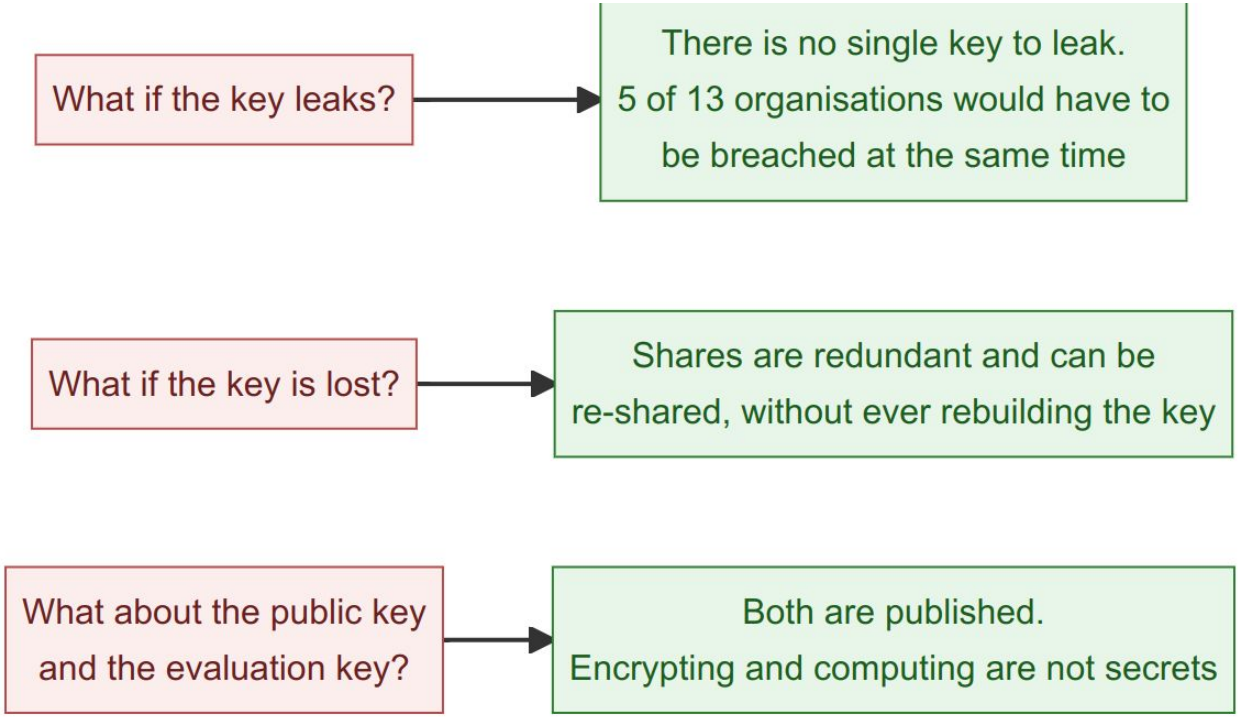
- They hold every encrypted value of the network, but can not read them

How Zama works: no one has the key

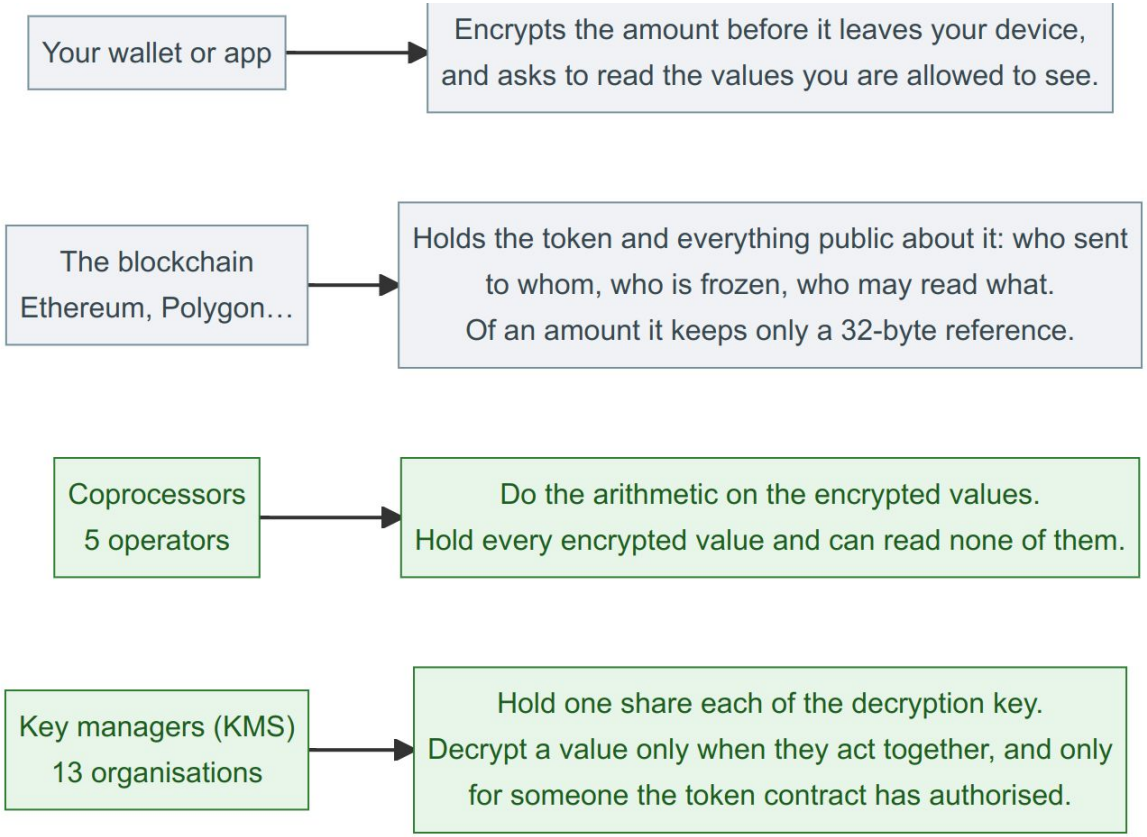
- The private key, the only one that could decrypt, is held by nobody;
    - it is secret-shared across 13 independent organisations, the Key Management Service (KMS).
- Decryption is a multi-party protocol:
    - Each KMS party computes a partial decryption with its own share and the parts are combined.
    - The key itself is never reassembled, on any machine, at any moment.

What if the key leaks, or is lost?

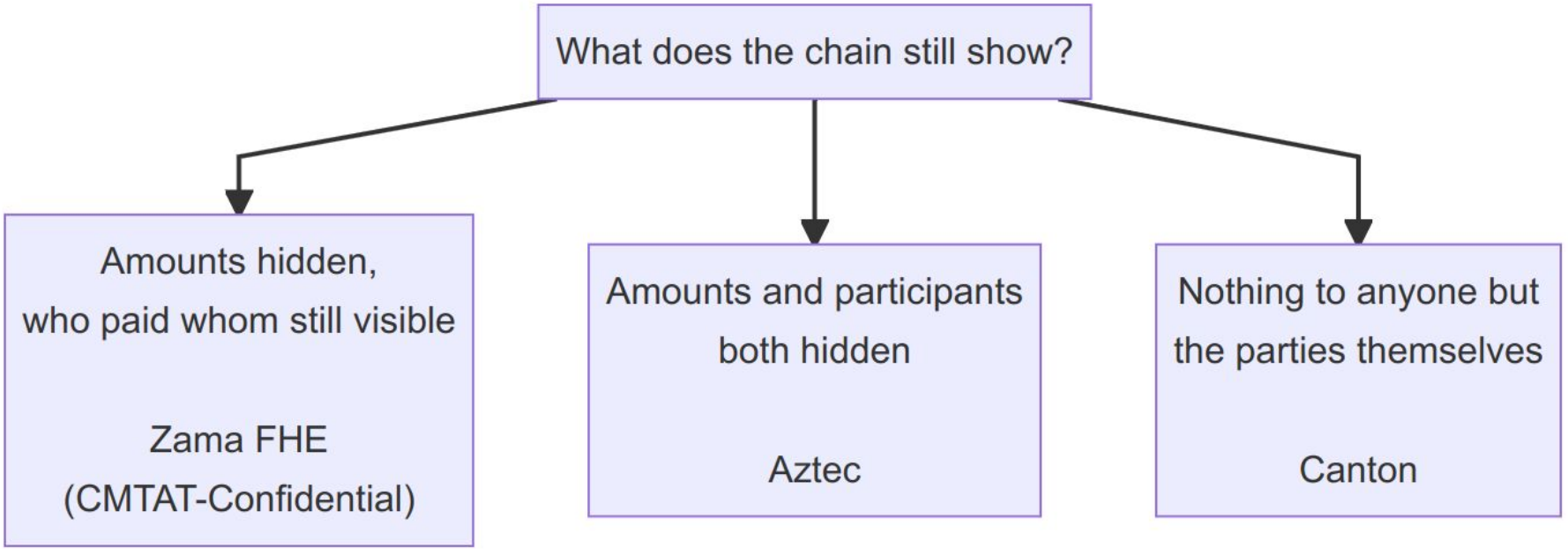
How Zama works: no one has the key



What does what

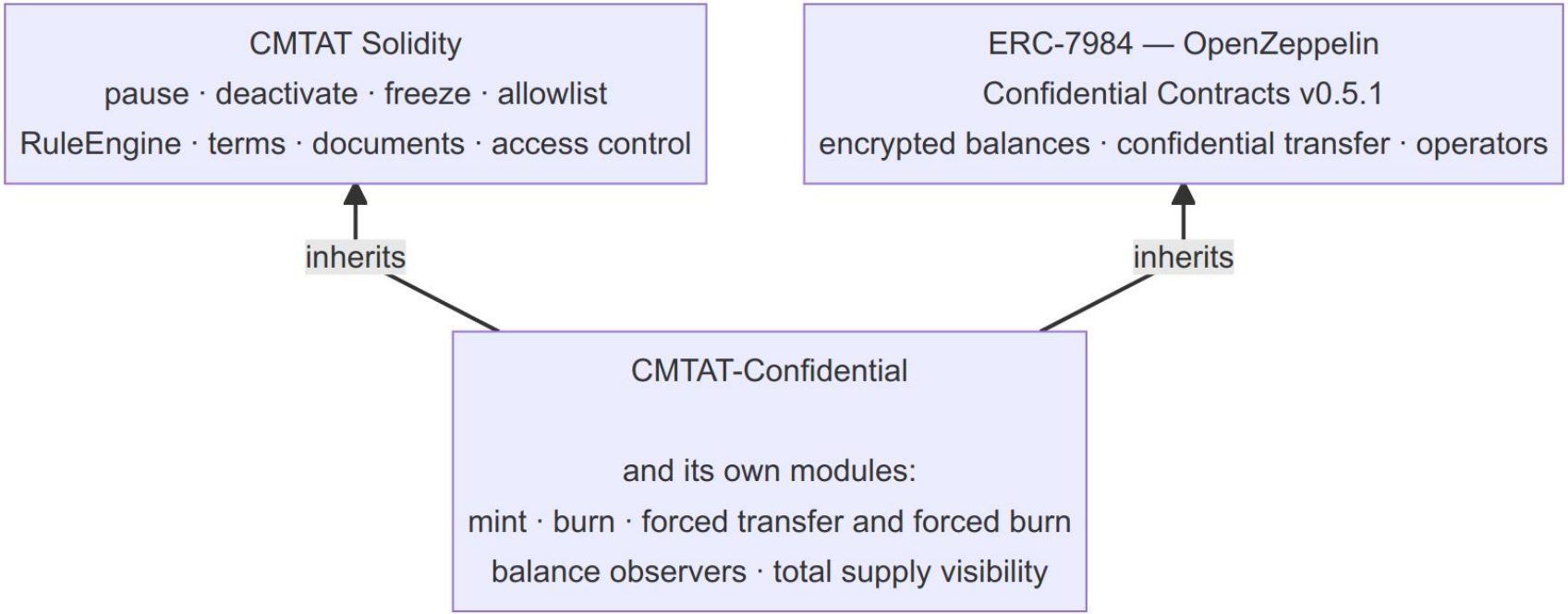


Difference with other privacy blockchains



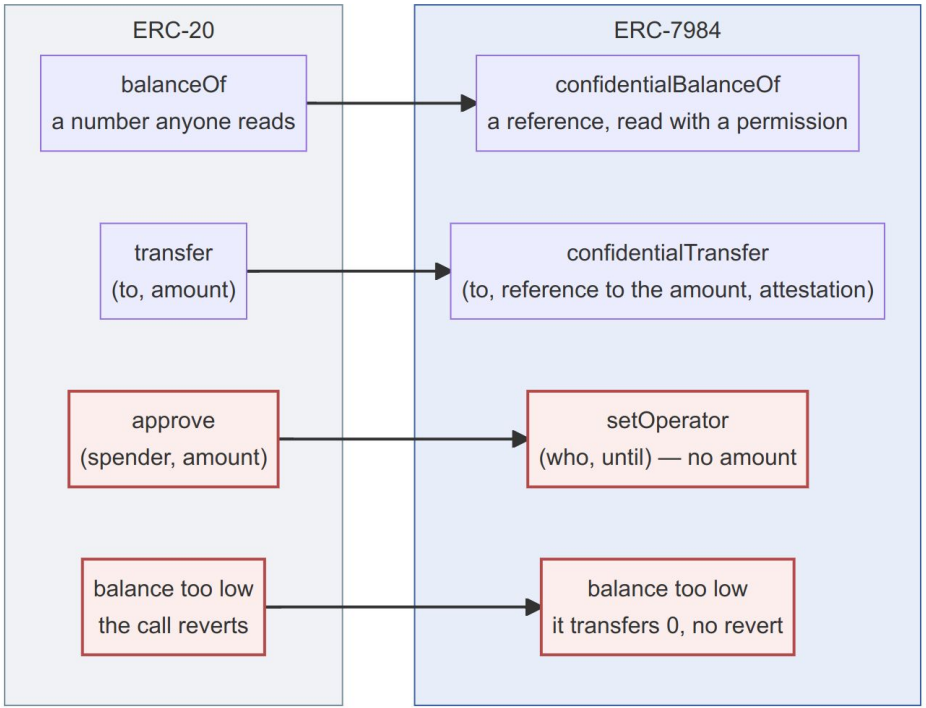
What the token is

CMTAT-Confidential building block

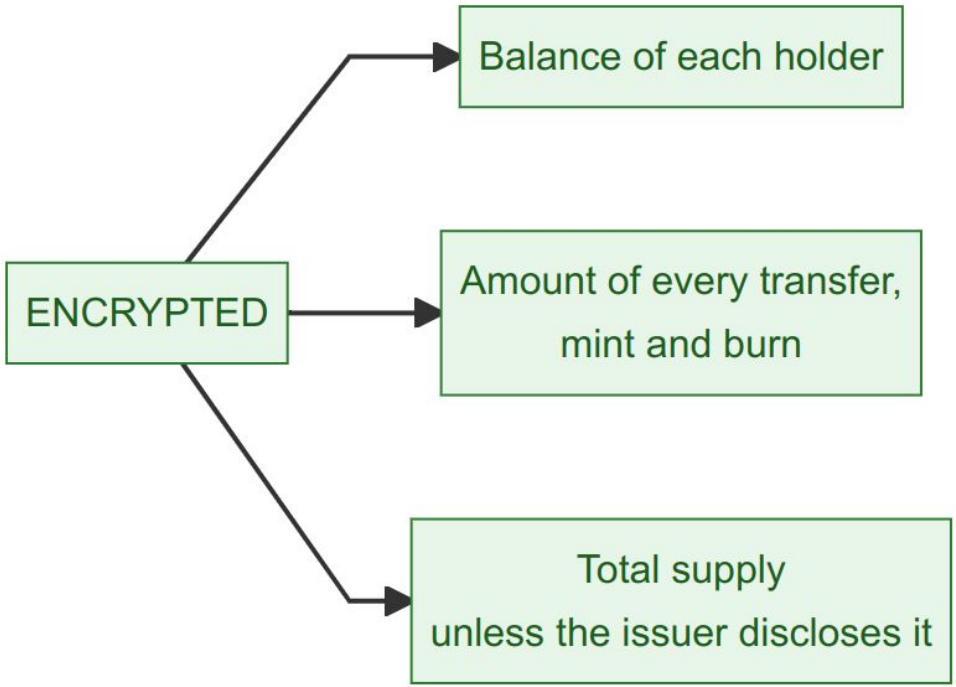


ERC-7984 Confidential ERC-20

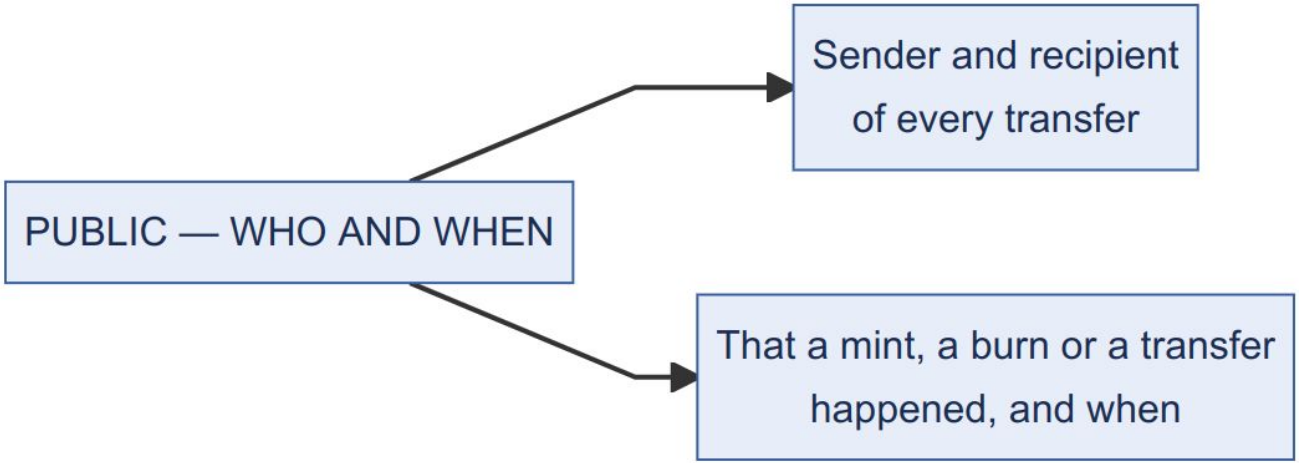
- ERC-7984 is the confidential counterpart of ERC-20;
- Drafted by OpenZeppelin with Zama;
- Reference implementation available on OpenZeppelin;
- CMTAT-Confidential is built on top of that implementation.



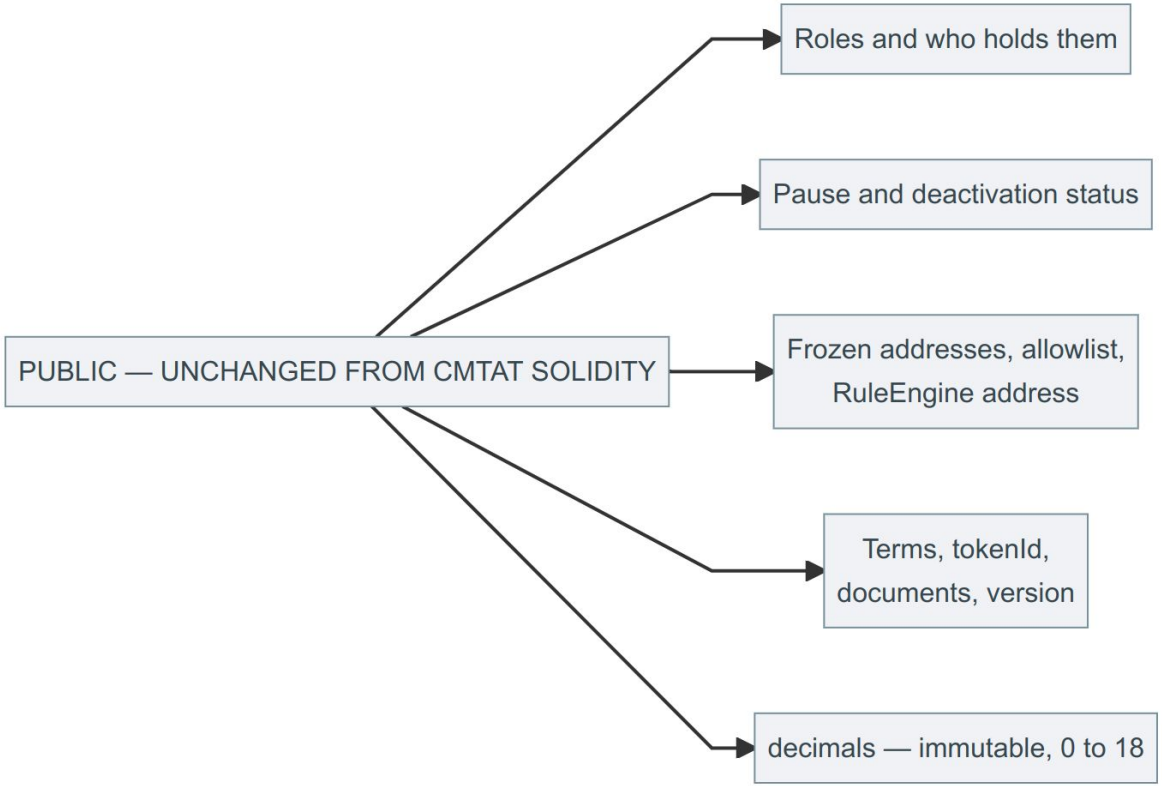
Encrypted: three elements



Public: who and when



Public: unchanged from CMTAT Solidity



Who can read

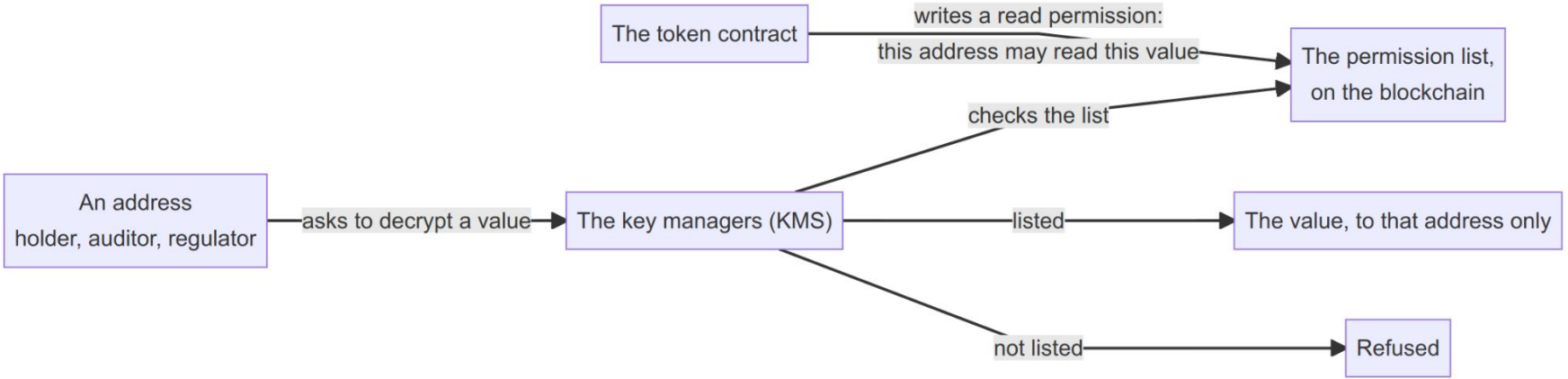
Who can read what and how

- Every actor uses an ordinary Ethereum ECDSA key pair;
- Reading an encrypted value requires a read permission recorded on-chain by the token contract;

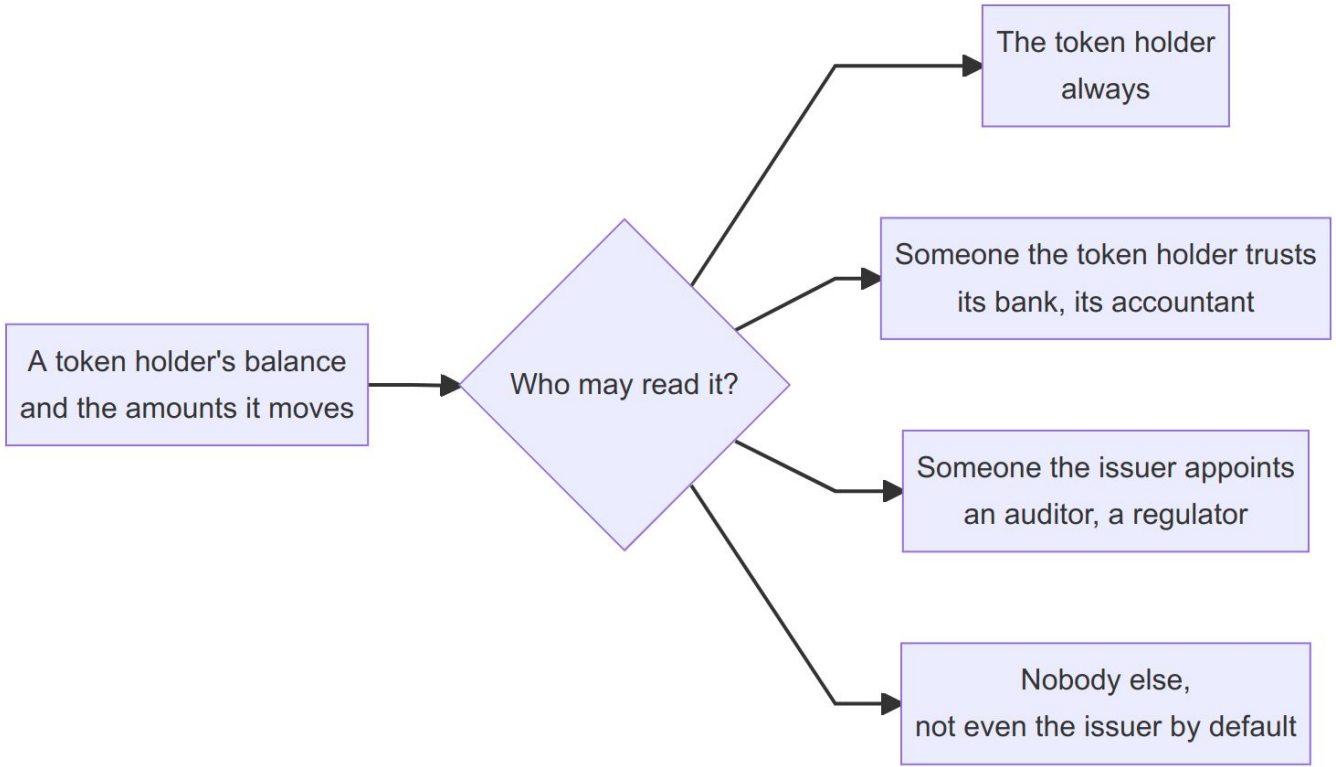
Handle (on-chain pointer)

- A permission is attached to one handle and is permanent: it cannot be revoked;
- Every operation produces a new handle;
- These permissions allow an issuer, an auditor or a regulator to request a decrypted version of the encrypted data (amount, total supply) from the KMS.

Who can read what and how



Observers - How it works



Total supply: private or public information

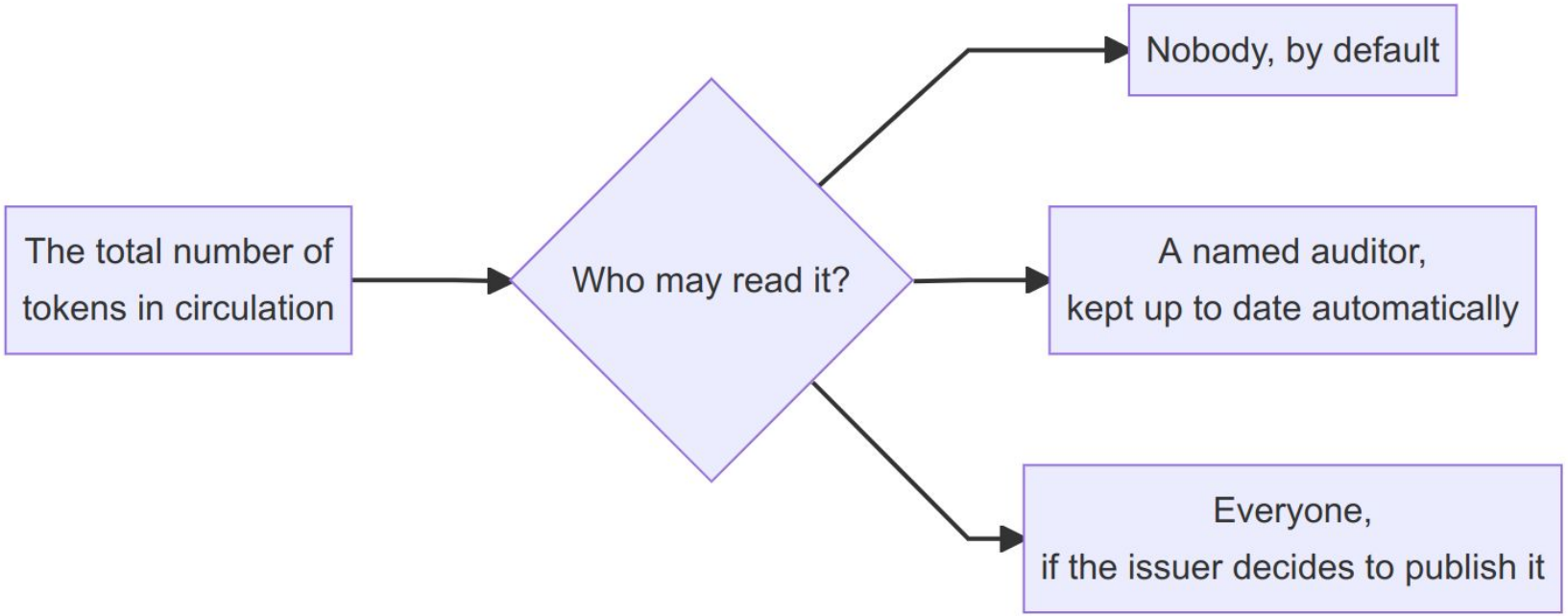
Why should the total supply be kept public?

- Total supply is required to compute the token value
- Information valuable for current token holders but future token holders if the token is available on the open market
- Problem: leak burn and mint amounts
- Alternative: keep the total supply private, restricted to the issuer and a named audit firm. The auditor is then responsible to attest publicly of the total supply

How it is implemented in CMTAT-Confidential

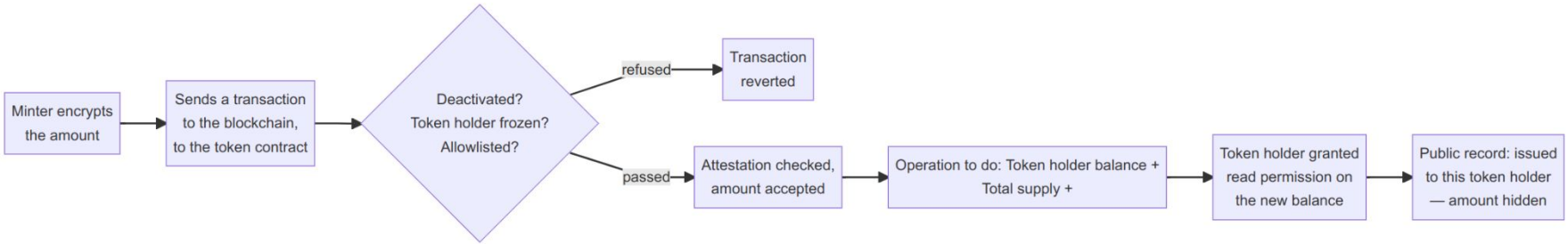
- Encrypted by default, in every deployment variant.
- The issuer may appoint named readers (an auditor, a regulator).
    - Their view is refreshed automatically after every mint and burn.
- Or publish it to everyone, once, with a transaction.

Total Supply Visibility

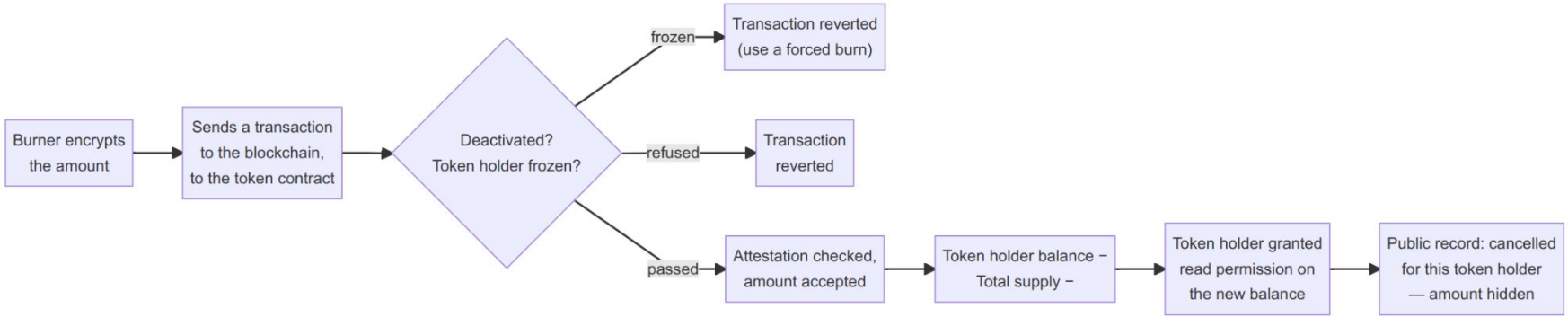


What it does

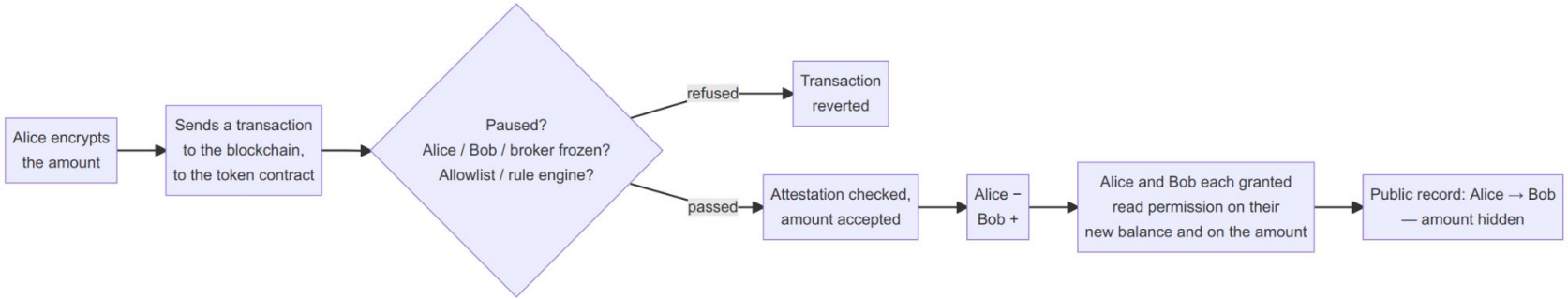
How a mint works



How a burn works

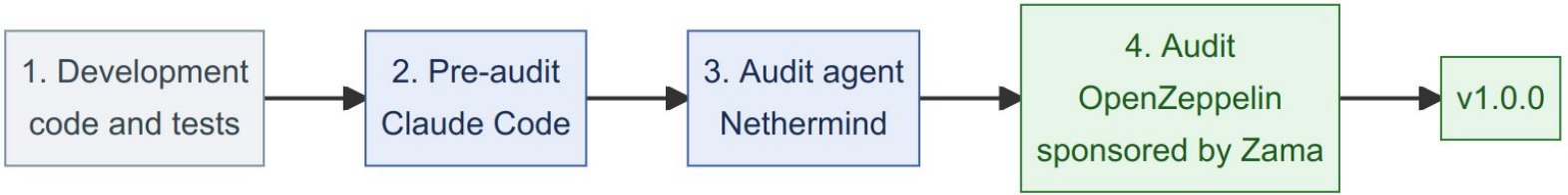


How a transfer works

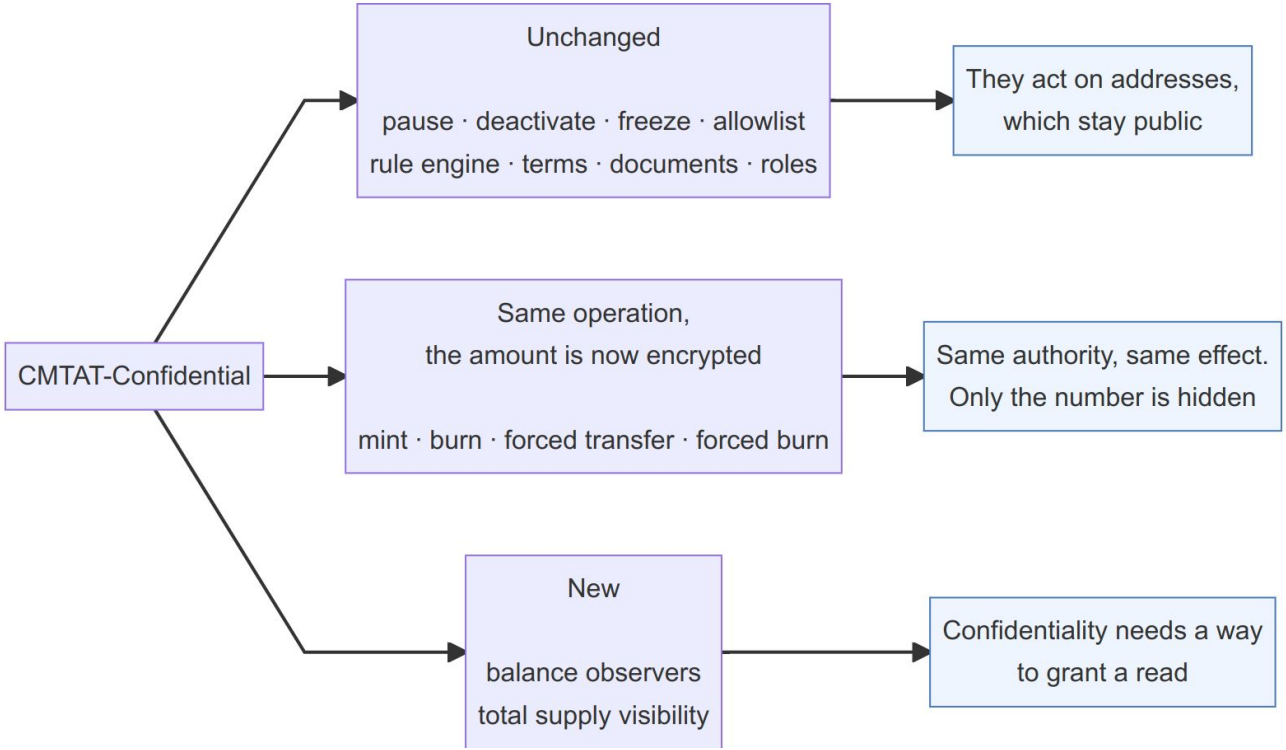


Closing

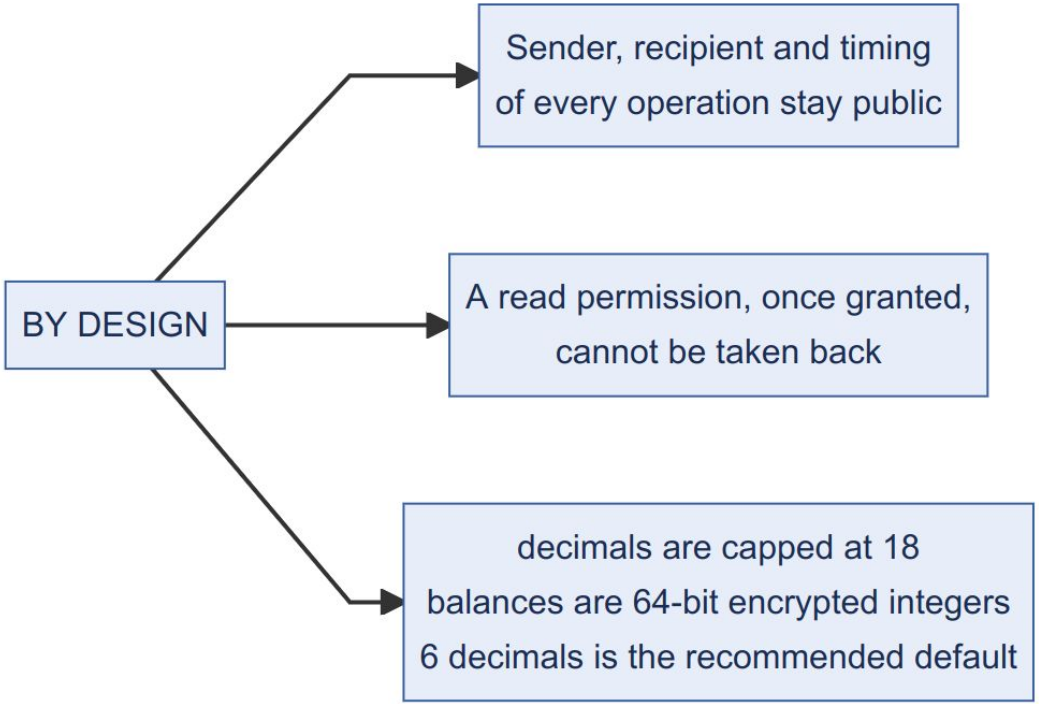
Development and review process



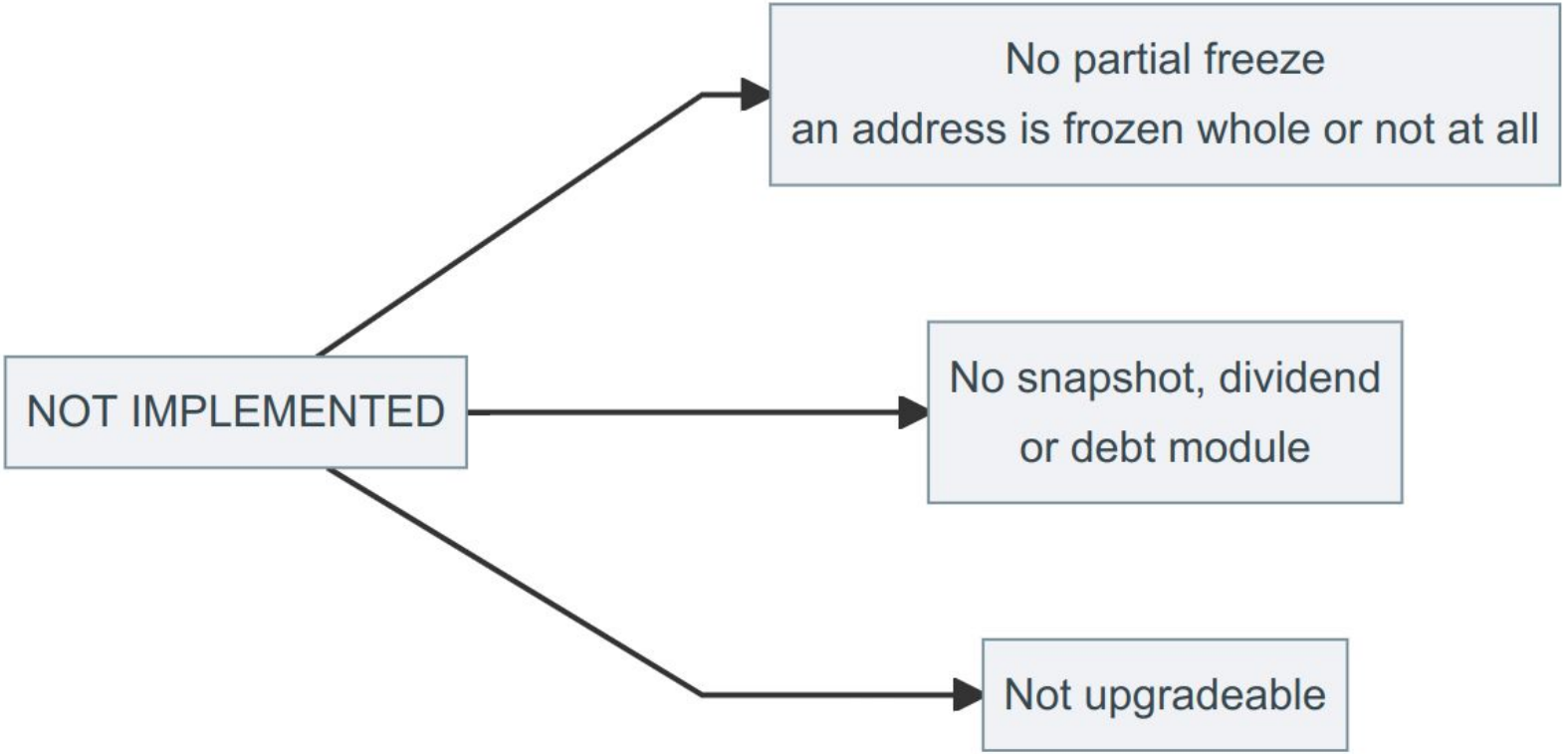
Conclusion


CMTAT-Confidential
Unchanged
pause · deactivate · freeze · allowlist
rule engine · terms · documents · roles
They act on addresses, which stay public
Same operation, the amount is now encrypted
mint · burn · forced transfer · forced burn
Same authority, same effect. Only the number is hidden
New
balance observers
total supply visibility
Confidentiality needs a way to grant a read

Limitations: by design



Limitations: not implemented



CMTAT-Confidential (GitHub)

Linkedin
(personal account)