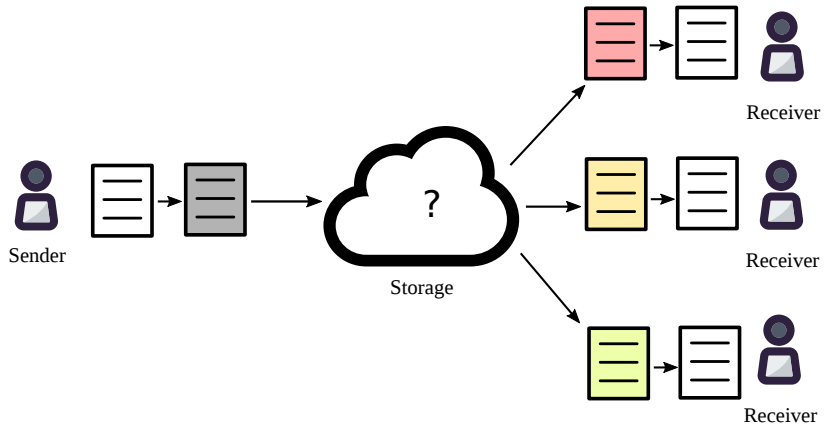


<fname lname>

<event>, <dd mmm yyyy>

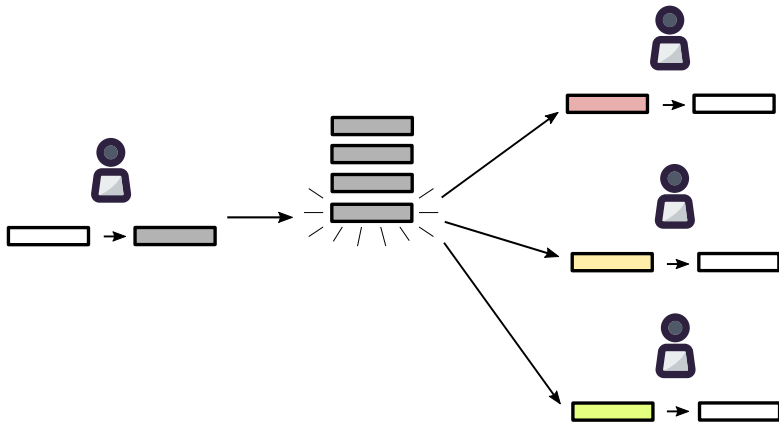
Why

Encrypted file sharing



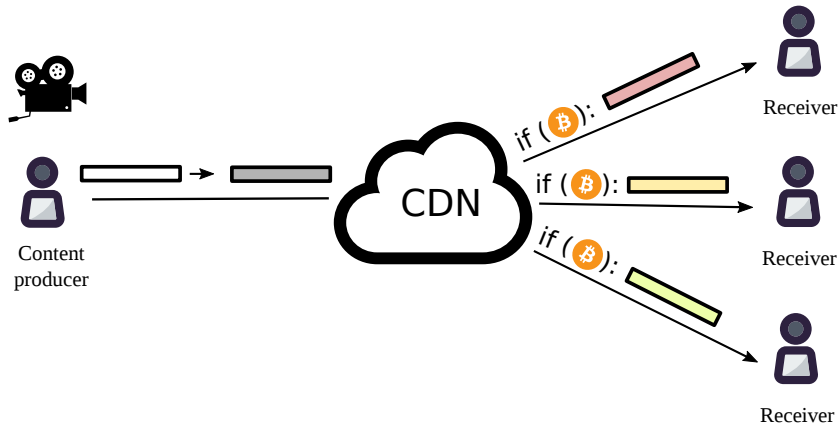
Why

Encrypted multi-user chats



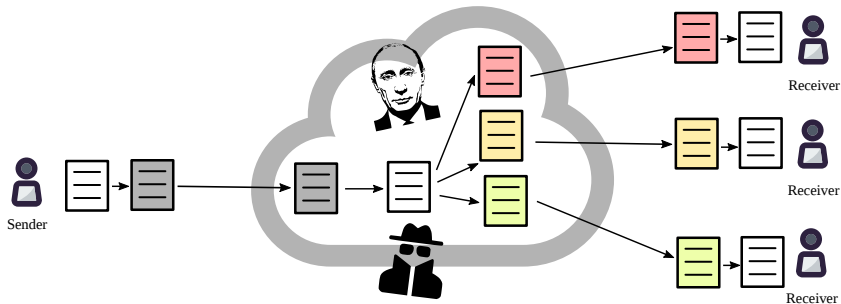
Why

Decentralized Netflix



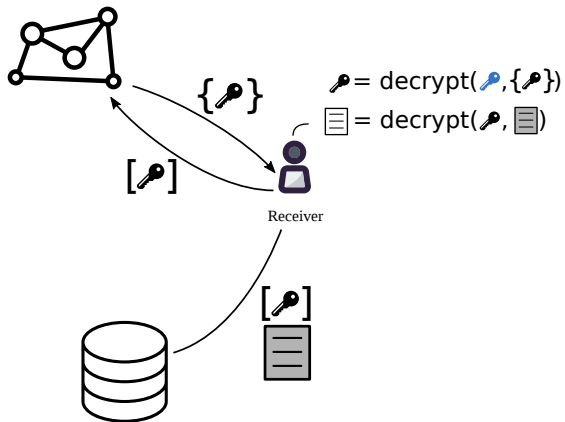
Central server + TLS

Data vulnerable to hackers, state actors etc

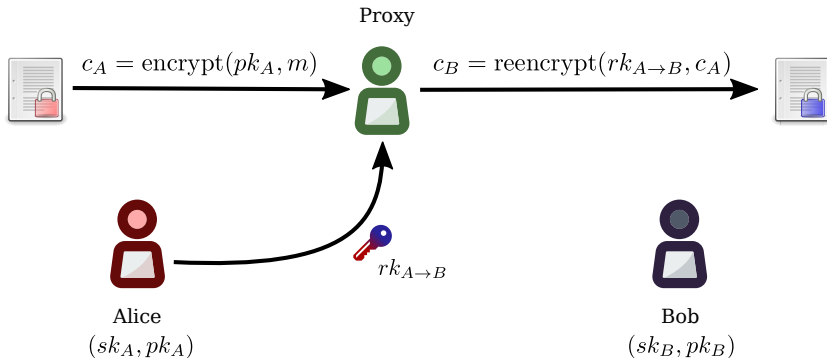


Solution

Proxy re-encryption + decentralization

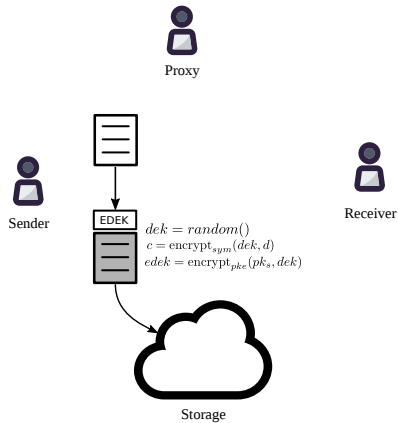


What is proxy re-encryption (PRE)



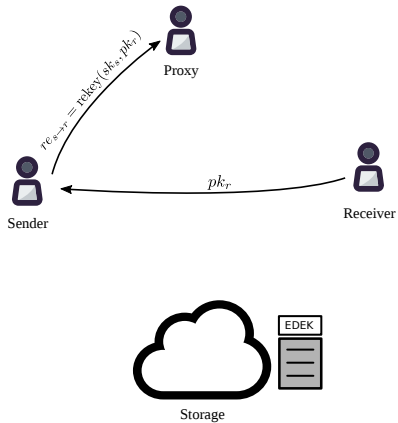
Centralized KMS using PRE

Encryption



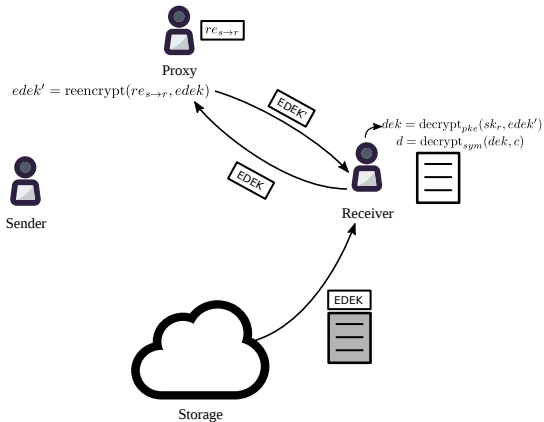
Centralized KMS using PRE

Access delegation



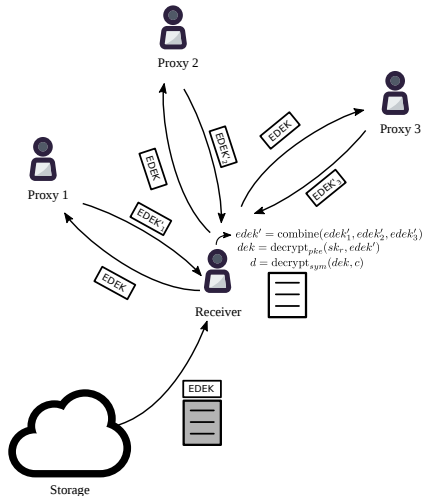
Centralized KMS using PRE

Decryption



Decentralized key management

Using threshold split-key re-encryption (Umbral)



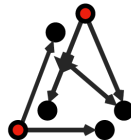
<https://github.com/nucypher/nucypher-kms/>

Umbral: threshold proxy re-encryption

- “Umbral” is Spanish for “threshold”
- PRE properties: Unidirectional, single-hop, non-interactive
- It follows a KEM/DEM approach:
 - ▶ UmbralKEM provides the threshold re-encryption capability
 - ▶ Uses ECIES for key encapsulation with zero knowledge proofs of correctness for verifiability on prime order curves (such as secp256k1)
 - ▶ The DEM can be any authenticated encryption (currently ChaCha20-Poly1305)
- IND-PRE-CCA security
- Verification of re-encryption correctness through Non-Interactive ZK Proofs
- Reference implementation: <https://github.com/nucypher/pyUmbral/>
- Documentation (WIP): <https://github.com/nucypher/umbral-doc>

Security Audits

pyUmbral, Cryptography, KMS Network, Smart Contracts



**Least
Authority**
Freedom Matters

PRE demo



Demo network: <https://github.com/nucypher/mock-net/>

NU token

Purpose

- Splitting trust between re-encryption nodes (more tokens = more trust and more work);
- Proof of Stake for minting new coins according to the mining schedule;
- Security deposit to be at stake against malicious behavior of nodes

NU token

Mining

Mining reward:

$$\kappa = \left(0.5 + 0.5 \frac{\min(T_i, T_1)}{T_1} \right) \quad (1)$$

$$T_{i,\text{initial}} \geq T_{\min}, \quad (2)$$

$$\delta s_{i,t} = \kappa \frac{l_i}{\sum l_j} \frac{\ln 2}{T_{1/2}} (s_{\max} - s_{t-1}). \quad (3)$$

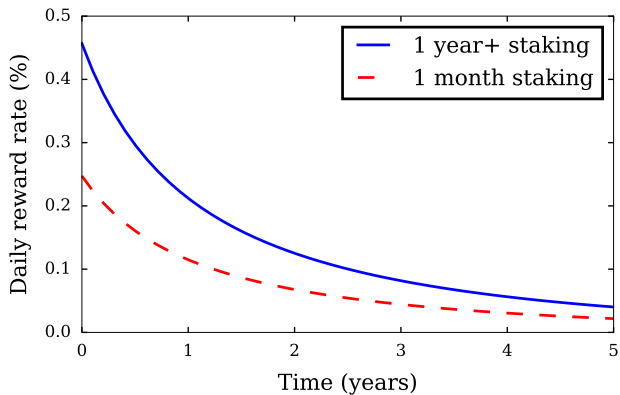
$$(4)$$

Results into:

$$\text{reward} \propto 2^{\frac{t}{T_{1/2}}}$$

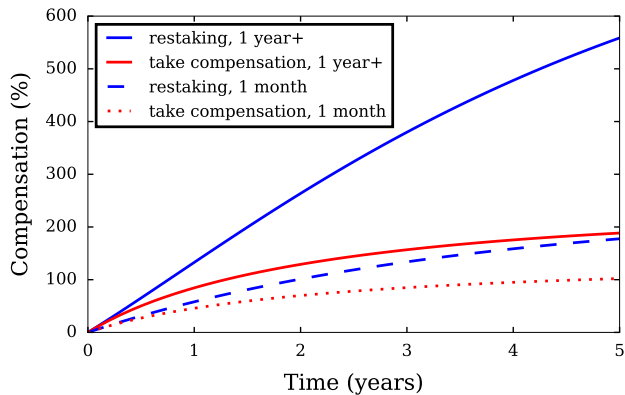
NU token

Graph of daily mining compensation



NU token

Relocking mining rewards



Usage examples

Decentralized marketplaces:

- Datum;
- Origin protocol;
- The Seam;
- SwipeCrypto.

Decentralized databases:

- Bluzelle;
- Fluence;
- Wolk.

Medical data sharing

- Medibloc;
- IRYO;

- Medixain;
- Wholesome;
- Medcredits;
- HealthCombix / PointNurse;
- Genobank;
- iku.network.

IoT

- Spherity (together with BigchainDB);
- Carblock.io;
- XAIN.

Cryptocurrency keys

- Coval Emblem Vault.

Types of policies

- Time-based;
- On payment (“grant access once paid, continue granting while paying”);
- Smart contract (public) method.

Open question

Is it possible to “grant to whoever pays”, without knowing public key, using non-interactive zero-knowledge proofs? (Performance of granting access is not required)

Useful links



Website: <https://nucypher.com>

Github: <https://github.com/nucypher/>

PyUmbral: <https://github.com/nucypher/pyUmbral/>

GoUmbral: <https://github.com/nucypher/goUmbral/>

Mocknet: <https://github.com/nucypher/mock-net/>

Discord: <https://discord.gg/7rmXa3S>

Whitepaper: <https://www.nucypher.com/whitepapers/english.pdf>

E-mail: <name>@nucypher.com

E-mail: hello@nucypher.com