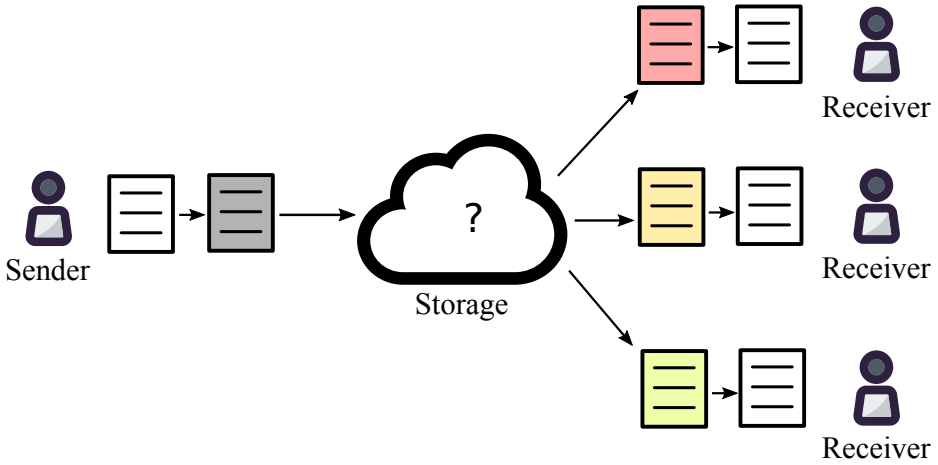
NuCypher

<fname lname>

<event>, <dd mmm yyyy>

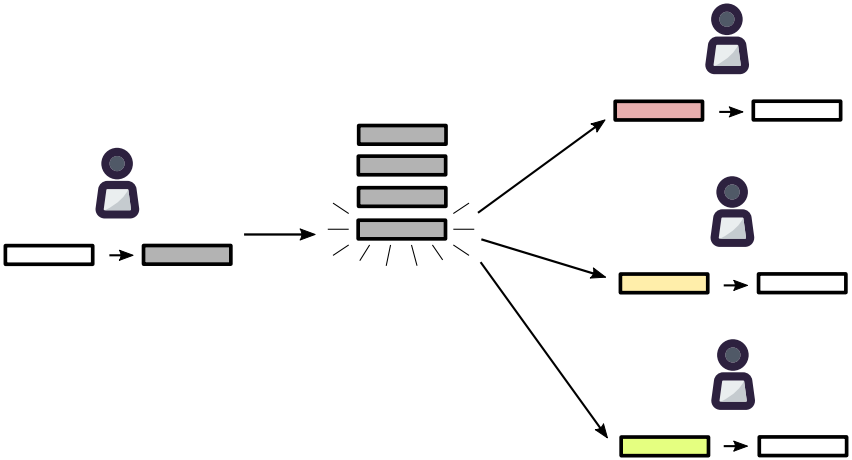
Why

Encrypted file sharing



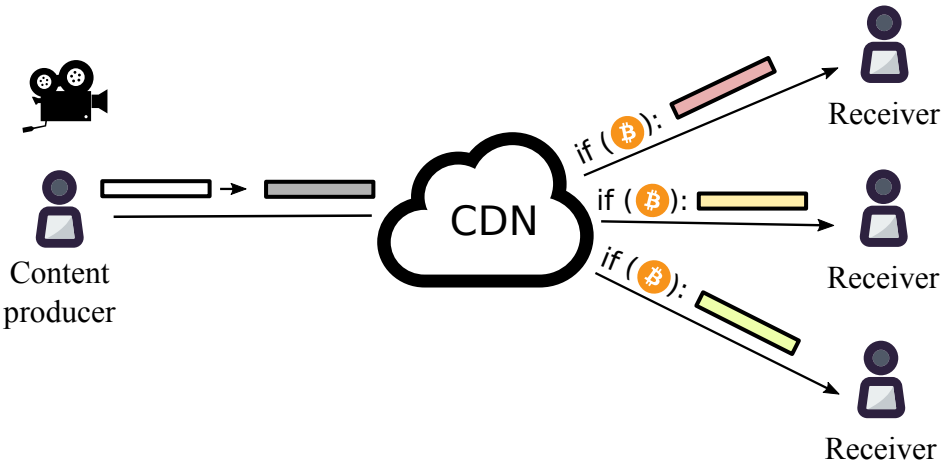
Why

Encrypted multi-user chats



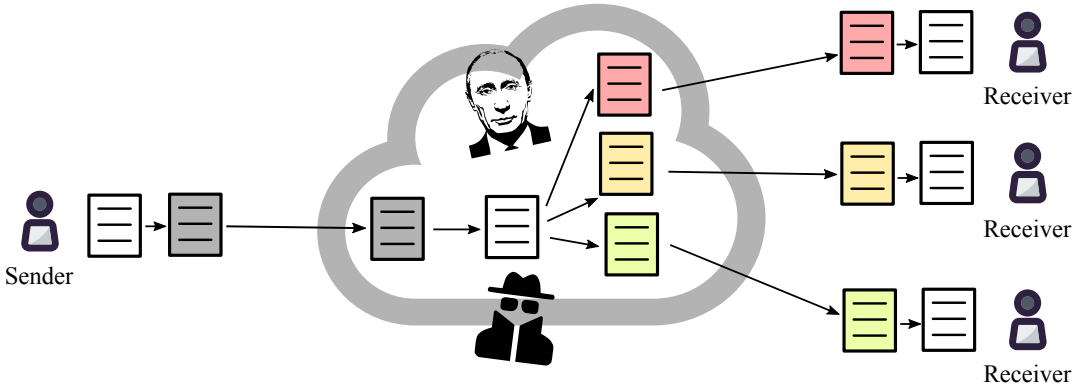
Why

Decentralized Netflix



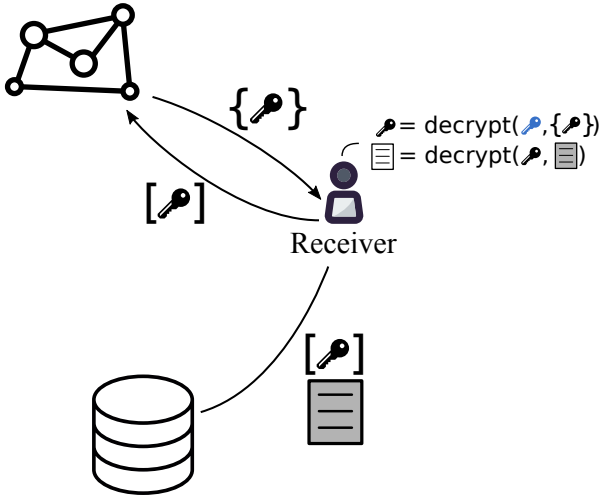
Central server + TLS

Data vulnerable to hackers, state actors etc

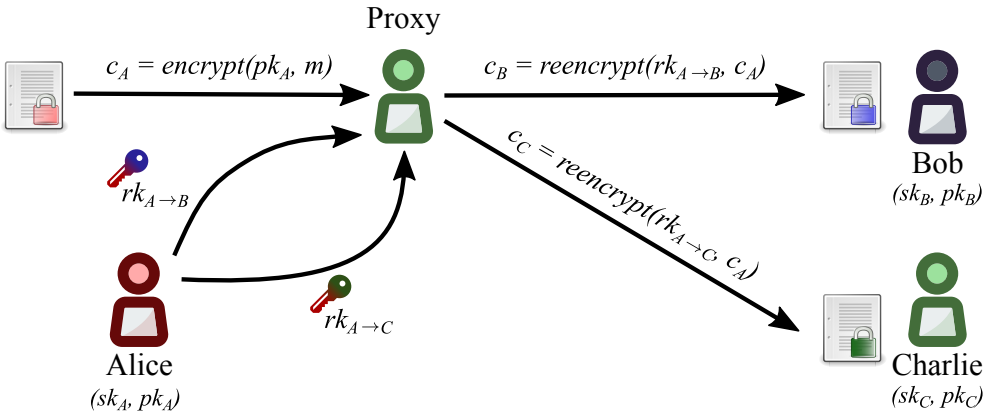


Solution

Proxy re-encryption + decentralization

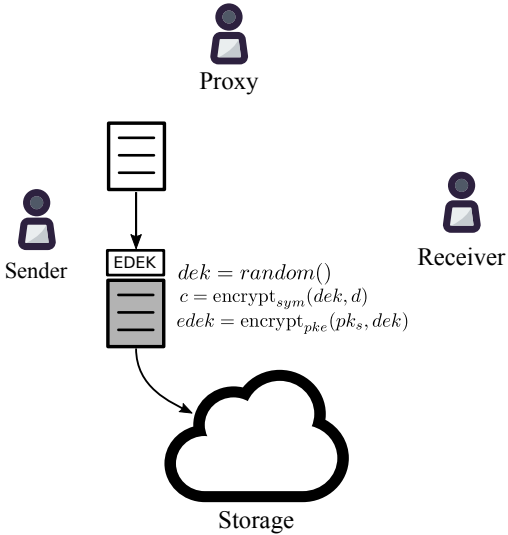


What is proxy re-encryption (PRE)



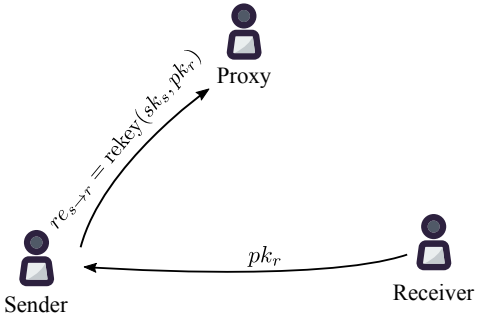
Centralized KMS using PRE

Encryption



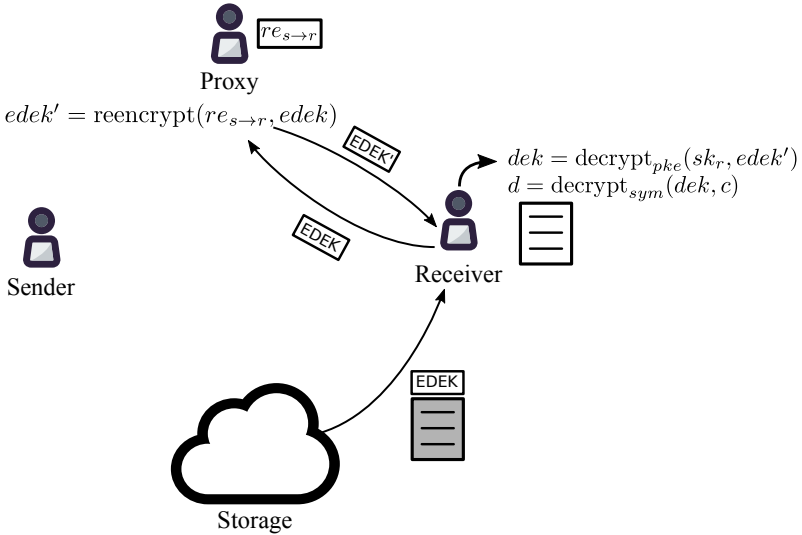
Centralized KMS using PRE

Access delegation



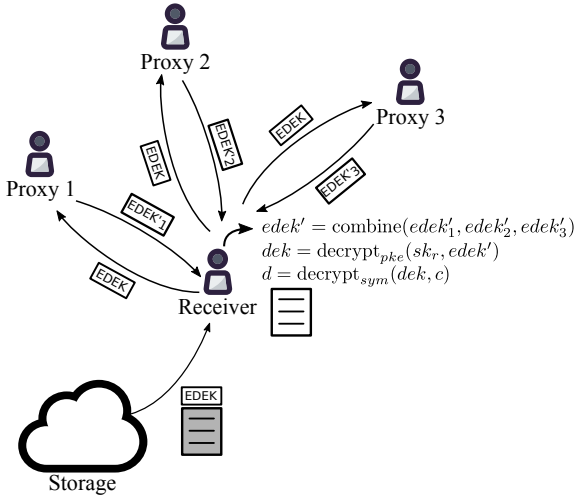
Centralized KMS using PRE

Decryption



Decentralized key management

Using threshold split-key re-encryption (Umbral)



https://github.com/nucypher/nucypher-kms/
https://github.com/nucypher/pyUmbral/

Types of policies

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

Open question

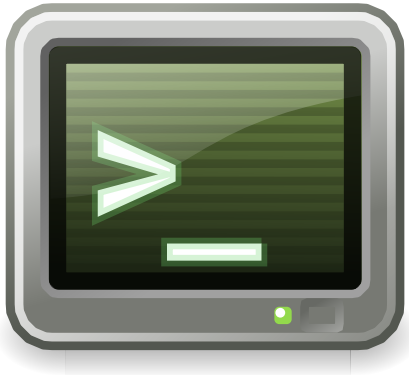
Is it possible to “grant to whoever pays”, without knowing public key, using non-interactive zero-knowledge proofs? (Performance of granting access is not required)

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

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:

$$\kappa = \left(0.5 + 0.5\frac{\min(\mathsf{T_i}, \mathsf{T}_1)}{\mathsf{T}_1}\right) \tag{1}$$

$$\mathsf{T_{i,initial}} \geq \mathsf{T}_{\min}, \tag{2}$$

$$\delta \mathsf{s_{i,t}} = \kappa \frac{\mathsf{l_i}}{\sum \mathsf{l_j}} \frac{\ln 2}{\mathsf{T}_{1/2}} \left(\mathsf{S}_{\max} - \mathsf{S_{t-1}}\right). \tag{3}$$

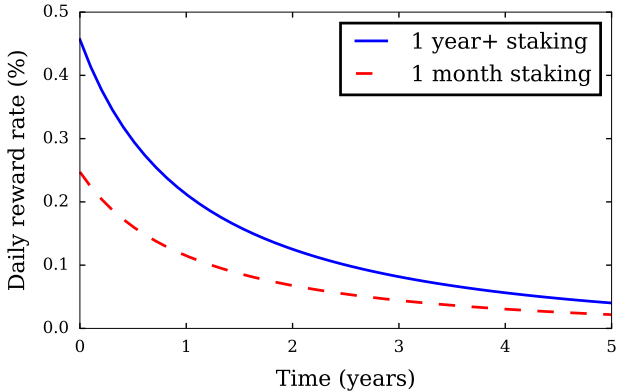
$$\tag{4}$$

Results into:

$$\text{reward} \propto 2^{\frac{\mathsf{t}}{\mathsf{T}_{1/2}}}$$

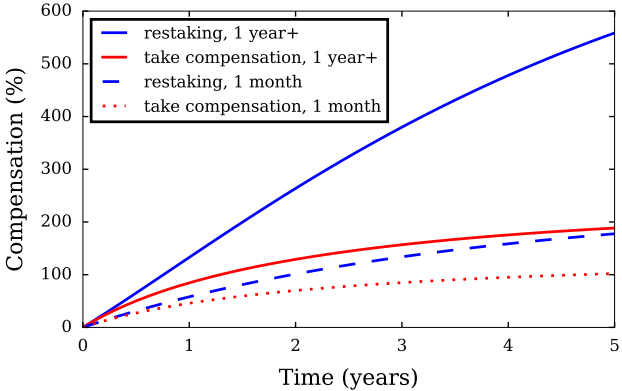
NU token

Graph of daily mining compensation

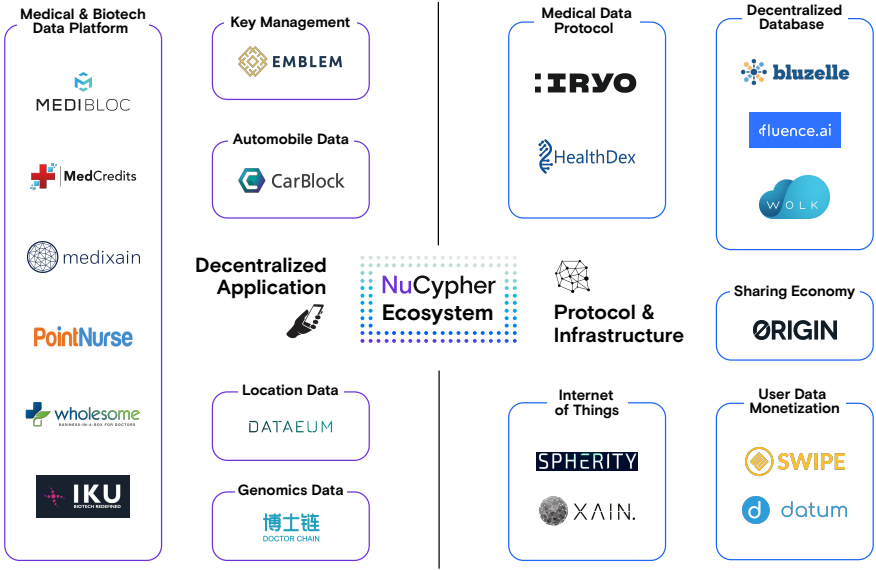


NU token

Relocking mining rewards



Early Users


Medical & Biotech Data Platform
MEDIBLOC
MedCredits
medixain
PointNurse
wholesome
IKU
Key Management
EMBLEM
Automobile Data
CarBlock
Location Data
DATAEUM
Genomics Data
博士链
DOCTOR CHAIN
Decentralized Application
NuCypher Ecosystem
Protocol & Infrastructure
Medical Data Protocol
IRYO
HealthDex
Internet of Things
SPHERITY
XAIN.
Decentralized Database
bluzelle
fluence.ai
WOLK
Sharing Economy
ØRIGIN
User Data Monetization
SWIPE
datum

Fully Homomorphic Encryption

nuFHE Library

- GPU implementation of fully homomorphic encryption
- Uses either FFT or integer NTT
- GitHub: `https://github.com/nucypher/nufhe`
- Achieved 100x performance over TFHE benchmarks

Platform	Library	Performance (ms/bit)	
		Binary Gate	MUX Gate
Single Core/Single GPU - FFT	TFHE (CPU)	13	26
	nuFHE	0.13	0.22
	Speedup	**100.9**	**117.7**
Single Core/Single GPU - NTT	cuFHE	0.35	N/A
	nuFHE	0.35	0.67
	Speedup	**1.0**	**-**

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`