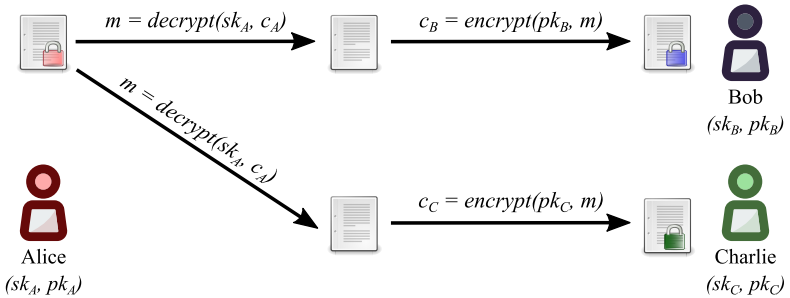
NuCypher

<presenter name(s), role(s)>

<event name>, <dd MMM yyyy>

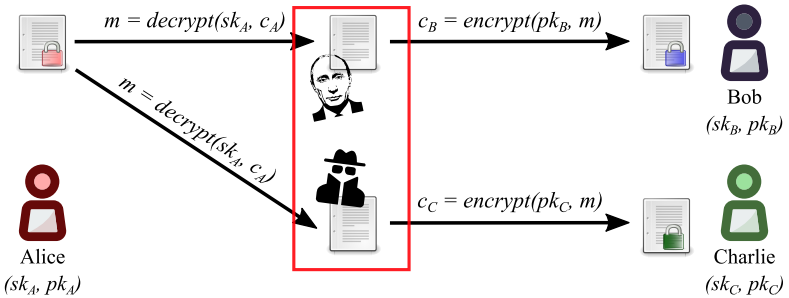
Public Key Encryption (PKE)



Limitations

- Decryption required before sharing
- Not scalable
- Complex access revocation

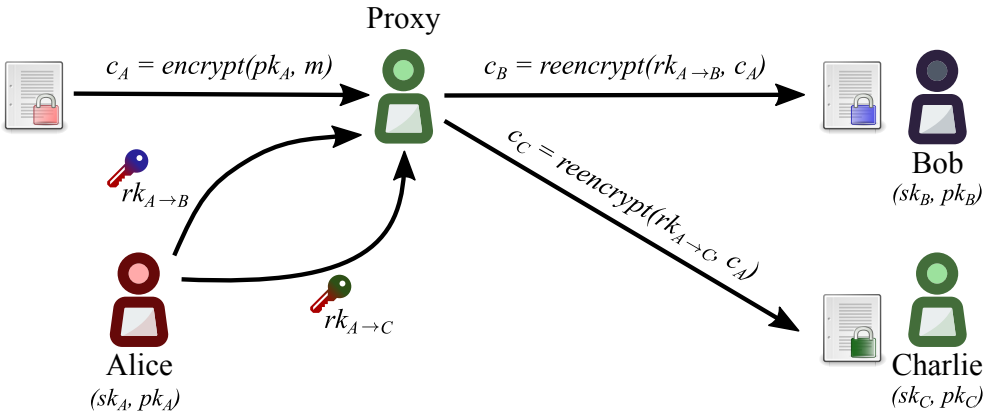
Public Key Encryption (PKE)



Limitations

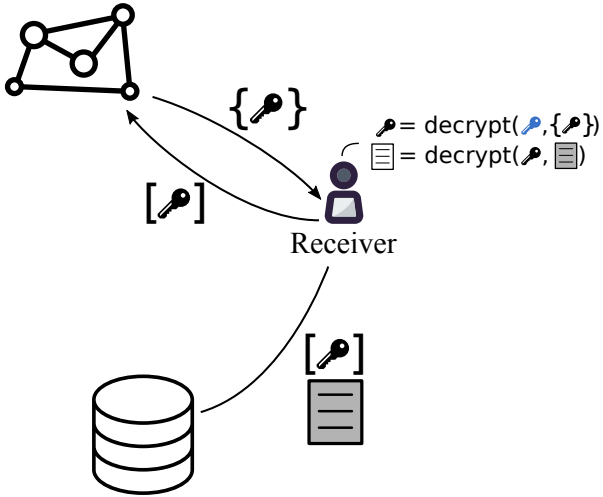
- Decryption required before sharing
- Not scalable
- Complex access revocation

What is proxy re-encryption (PRE)



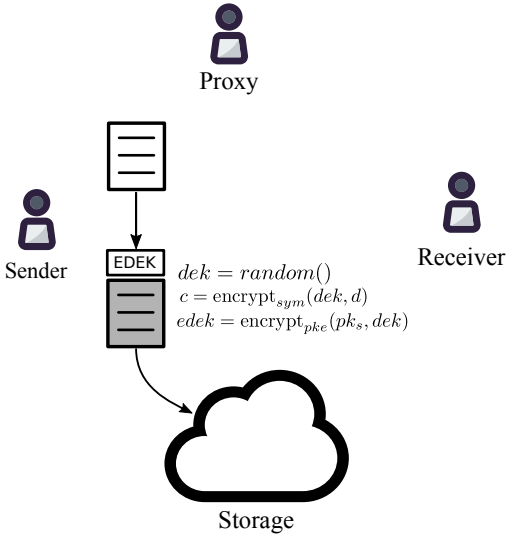
Solution

Proxy re-encryption + Key Management



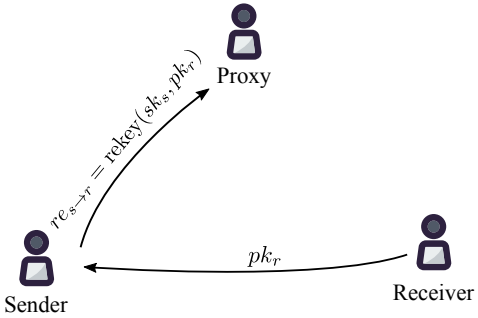
Centralized KMS using PRE

Encryption



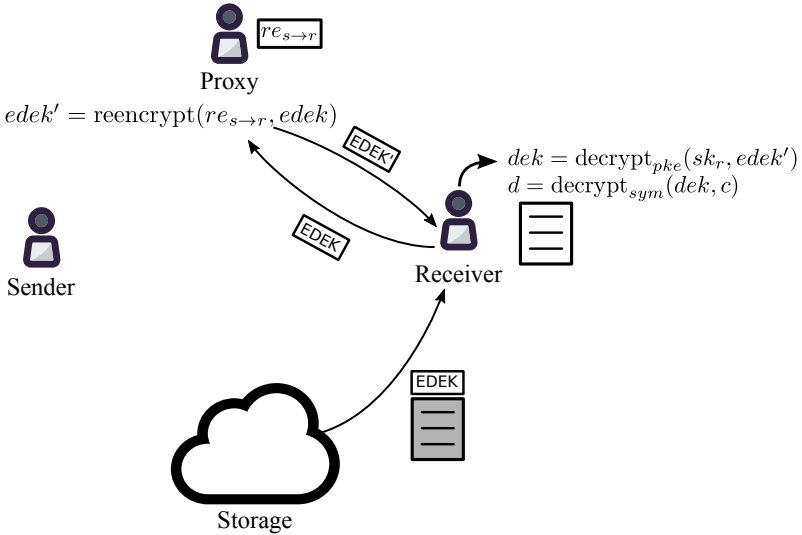
Centralized KMS using PRE

Access delegation



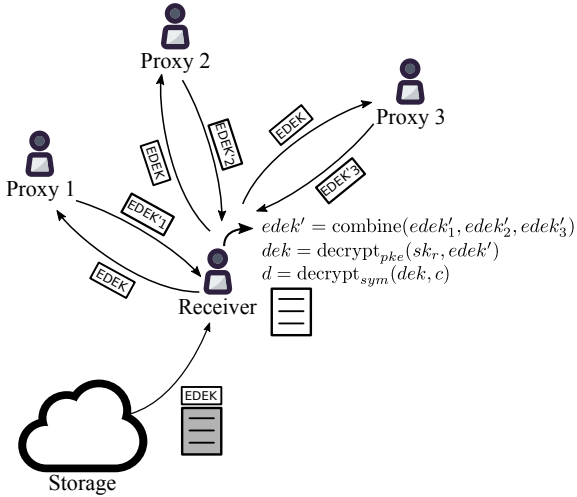
Centralized KMS using PRE

Decryption



Decentralized Key Management

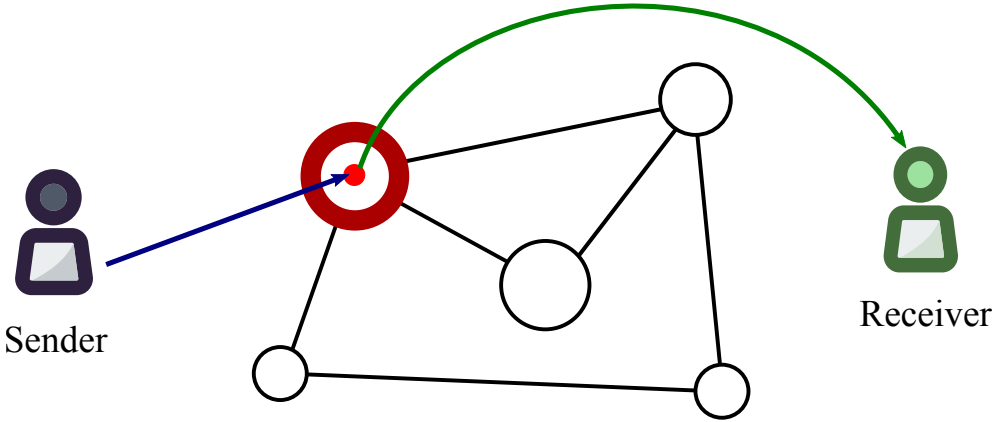
Using threshold split-key re-encryption (Umbral)



Umbral: Threshold Proxy Re-encryption

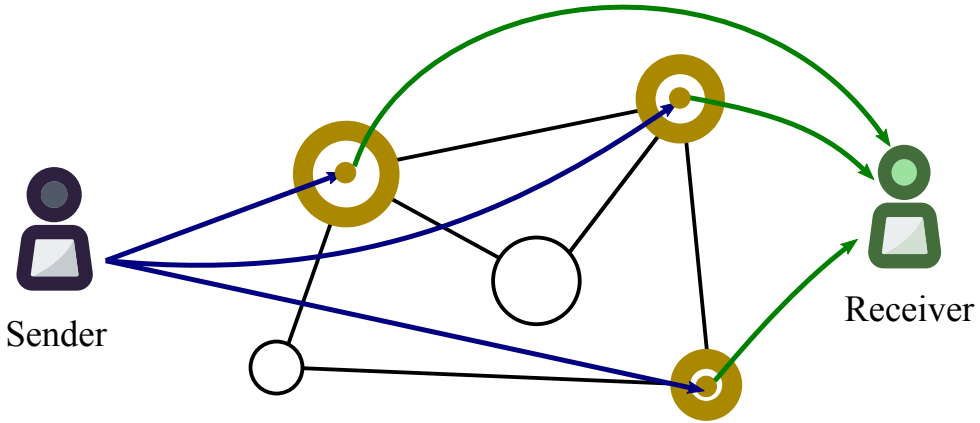
- “Umbral” is Spanish for “threshold”
- PRE properties: Unidirectional, single-hop, non-interactive
- It follows a KEM/DEM approach:
  - UmbralKEM provides the threshold re-encryption capability
  - Uses ECIES for key encapsulation with zero knowledge proofs of correctness for verifiability on prime order curves (such as secp256k1)
  - The DEM can be any authenticated encryption (currently ChaCha20-Poly1305)
- IND-PRE-CCA security
- Key splitting is analogous to Shamir Secret Sharing
- 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`

KMS Network: Data Sharing + PKE



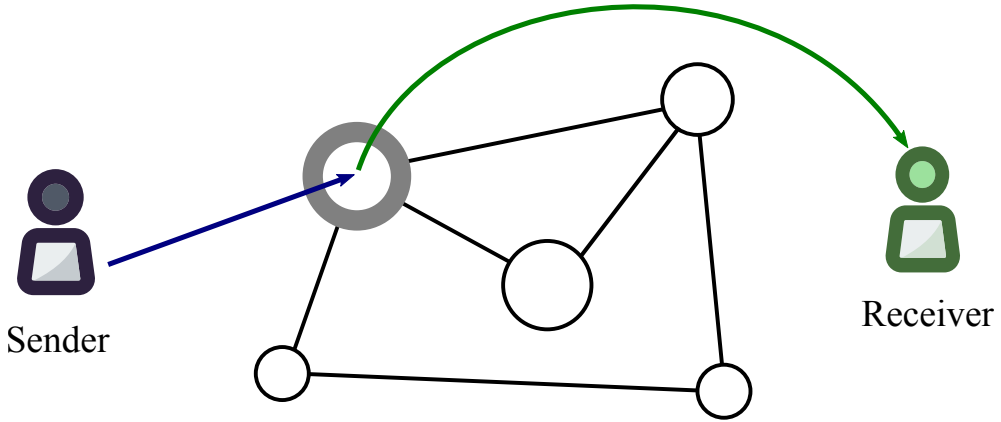
- Single node has access to data
- Single node can deny to do work

KMS Network: Data Sharing + PKE + Shamir Secret Sharing



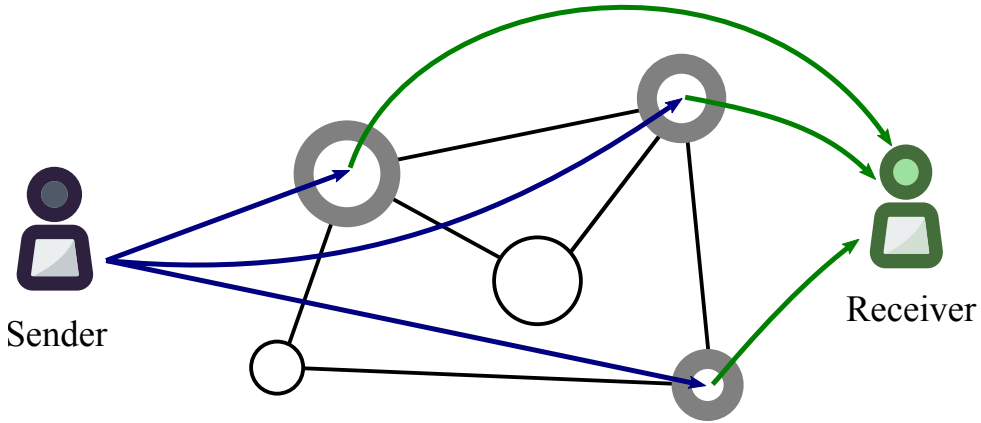
- Nodes can collude to gain access to data

KMS Network: Data Sharing + PRE



- Single node collusion with receiver possible
- Single node can deny to do work.

KMS Network: Data Sharing + Threshold PRE (Umbral)



- Collusion now requires m nodes + receiver

Data Sharing Policies

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

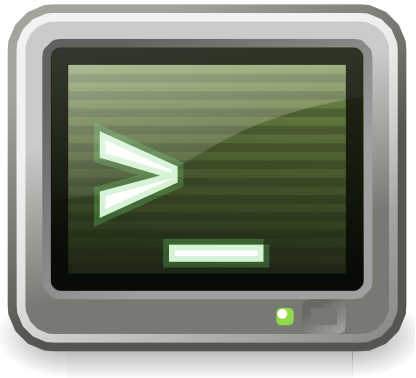
Decentralized proxies (Ursulas) are trusted to apply conditions without decrypting data

NU Token

Purpose

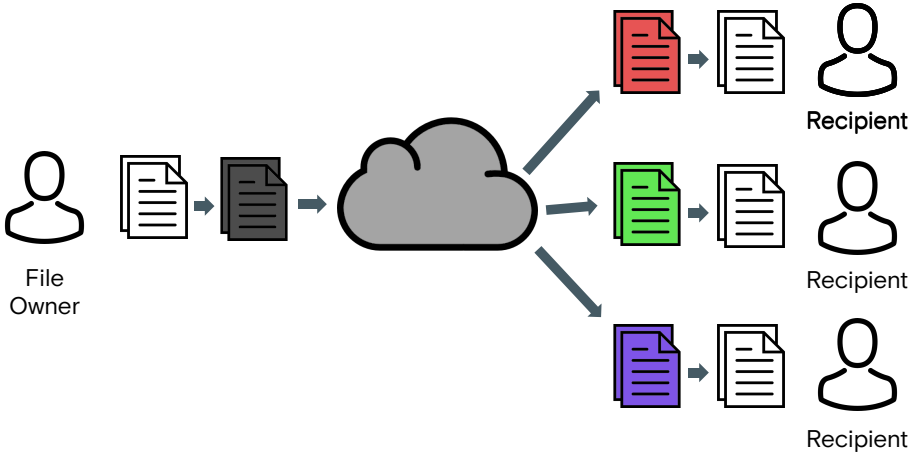
- Splitting trust between re-encryption nodes (more tokens = more trust and more work)
- Proof of Stake for minting new coins according to the mining schedule
- Security deposit to be at stake against malicious behavior of nodes

Demo



Use Cases

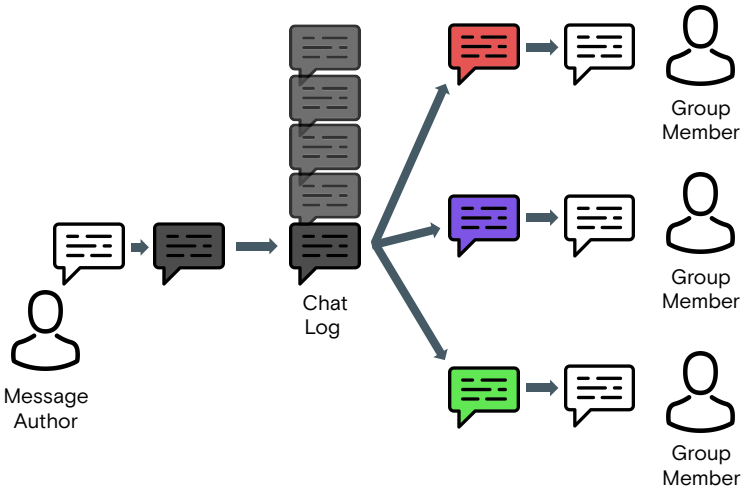
Encrypted file sharing



1. Encrypt → 2. Re-Encrypt → 3. Decrypt

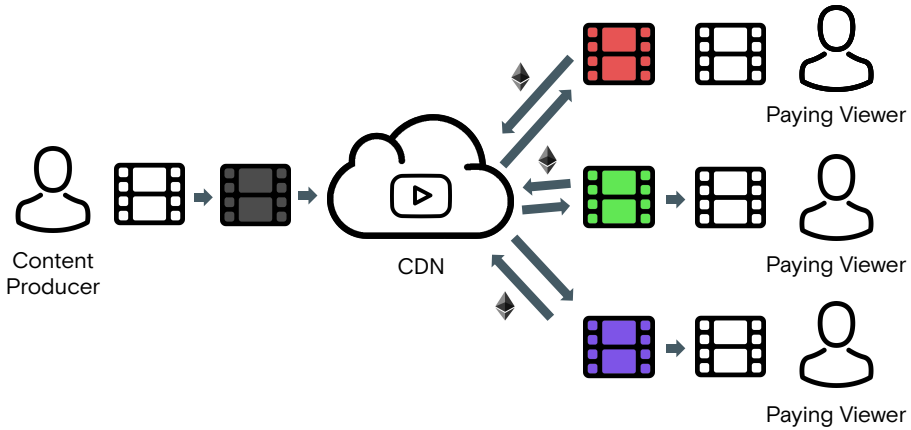
Use Cases

Encrypted multi-user chats



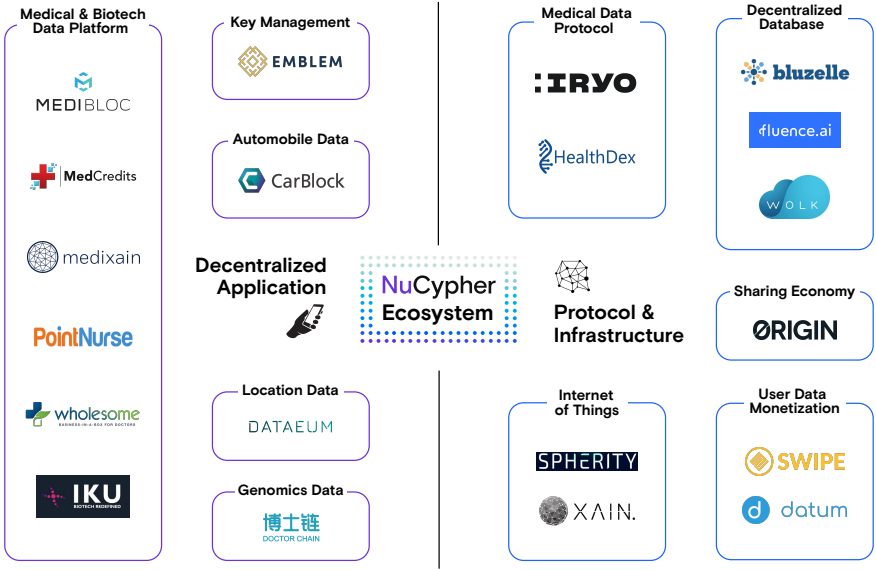
Use Cases

Decentralized Access-Controlled Content



1. Encrypt → **2. Conditional Re-Encrypt** → **3. Decrypt**

Early Users


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

Competing Technology

Data Masking and Tokenization

- Less secure for data with underlying patterns
- Reduce the value of data by obfuscating it

Multi-Party Computation

- Interactive protocol
- Slow Performance

Fully Homomorphic Encryption

- Slow Peformance
  - NuCypher is investing efforts in this area

Fully Homomorphic Encryption

nuFHE Library

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

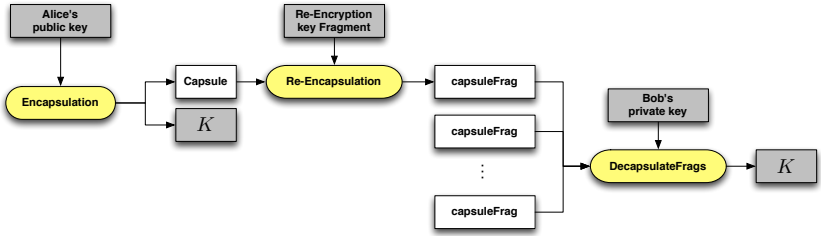
Platform	Library	Performance (ms/bit)	
		Binary Gate	MUX Gate
Single Core/Single GPU - FFT	TFHE (CPU)	13	26
	nuFHE	0.13	0.22
	Speedup	**100.9**	**117.7**
Single Core/Single GPU - NTT	cuFHE	0.35	N/A
	nuFHE	0.35	0.67
	Speedup	**1.0**	**-**

More Information

NuCypher

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: <email>@nucypher.com
E-mail: hello@nucypher.com

Appendix: Umbral Flow Diagram



- Reference implementation: `https://github.com/nucypher/pyUmbral/`
- Documentation: `https://github.com/nucypher/umbral-doc`

Appendix: Security Audits

Appendix: NU Token Metrics

Mining

Mining reward:

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

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

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

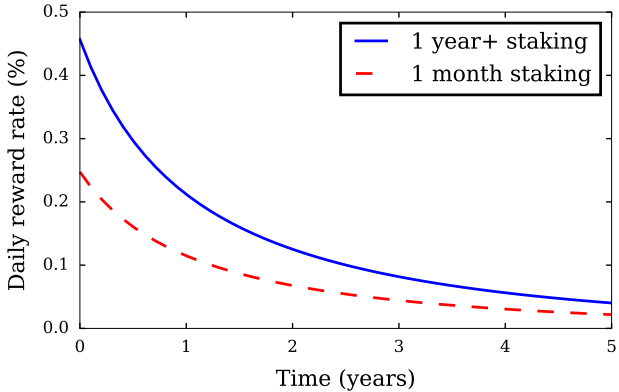
$$\tag{4}$$

Results into:

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

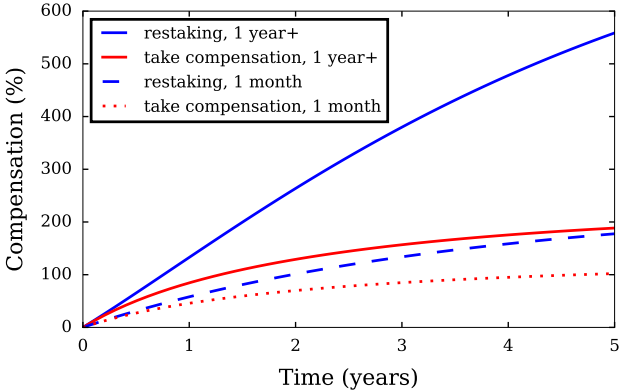
Appendix: NU Token Metrics

Graph of daily mining compensation



Appendix: NU Token Metrics

Relocking mining rewards



Appendix: Team

Founders

MacLane Wilkison
Co-founder and CEO

Michael Egorov, PhD
Co-founder and CTO

Employees

David Nuñez, PhD
Cryptographer

John Pacific (tux)
Engineer

Justin Myles Holmes
Engineer

Sergey Zotov
Engineer

Kieran Prasch
Engineer

Bogdan Opanchuk, PhD
Engineer

Ryan Caruso
Community

Derek Pierre
Business Development

Arjun Hassard
Product & Partnerships

Advisors

Prof. Dave Evans

Prof. Giuseppe Ateniese
Stevens Inst. of Technology

John Bantleman
Rainstor

Tony Bishop
Equinix