NuCypher

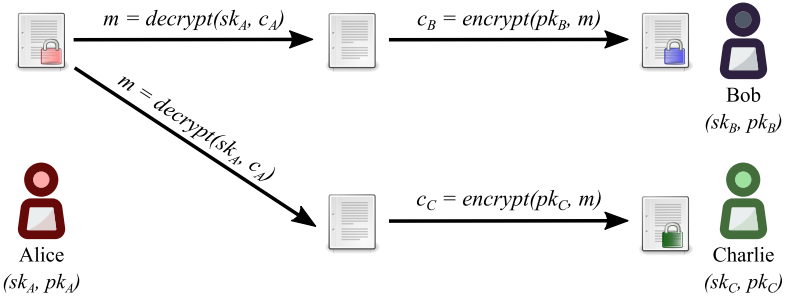
Derek Pierre, Biz Dev

ETH UofT, 08 Mar 2019

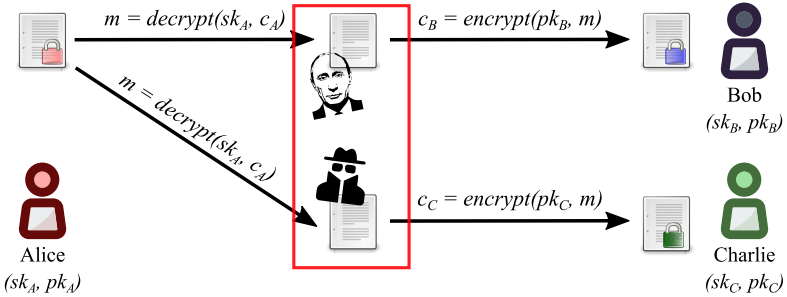
NuCypher Overview

- Use cryptography to build the tools & infrastructure to preserve data privacy
- Privacy-preserving solutions for distributed applications
  - Proxy Re-encryption (PRE)
    - Secure data-sharing and access control of encrypted data
  - Fully Homomorphic Encryption (FHE)
    - Perform arbitrary operations on encrypted data
- Blockchain & Private Deployments

Public Key Encryption (PKE)

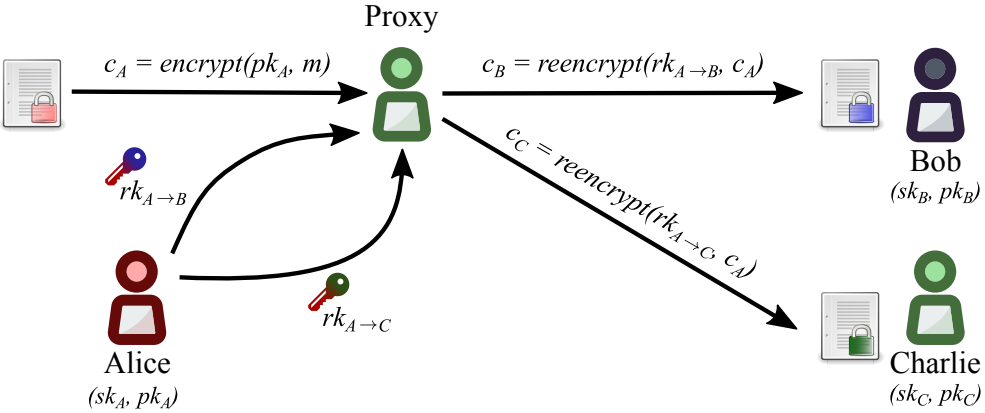


Public Key Encryption (PKE)



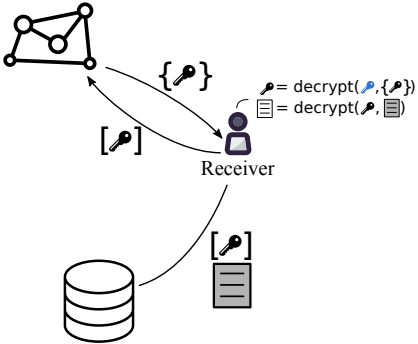
- Decryption required before sharing
- Not scalable
- Complex access revocation

What is proxy re-encryption (PRE)



Solution

Proxy re-encryption + Key Management

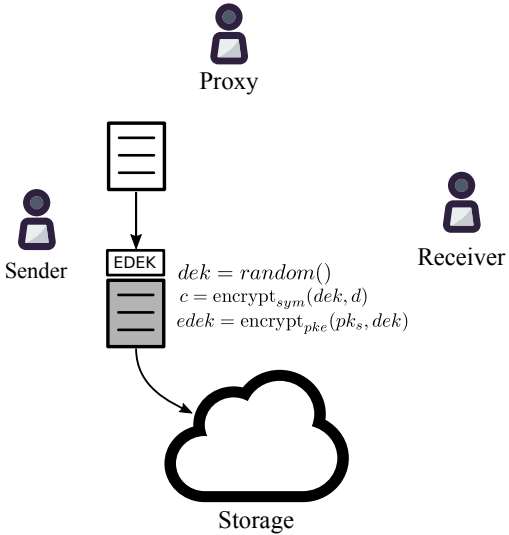


Advantages

- Data not decrypted to facilitate sharing
- Scalable and performant
- Access revocation through re-encryption key deletion
- Secure use of data storage providers

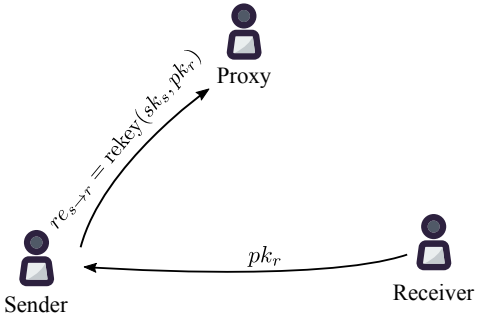
Centralized KMS using PRE

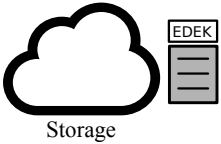
Encryption



Centralized KMS using PRE

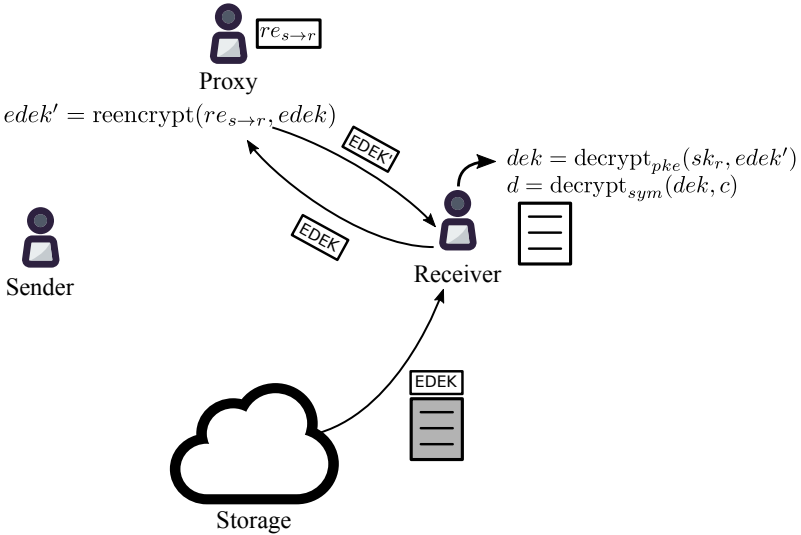
Access delegation





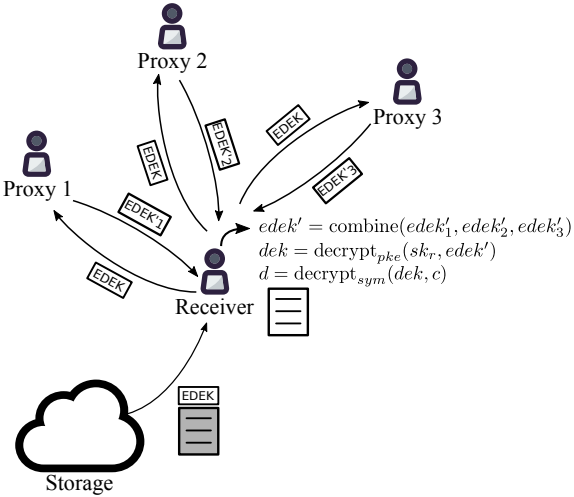
Centralized KMS using PRE

Decryption

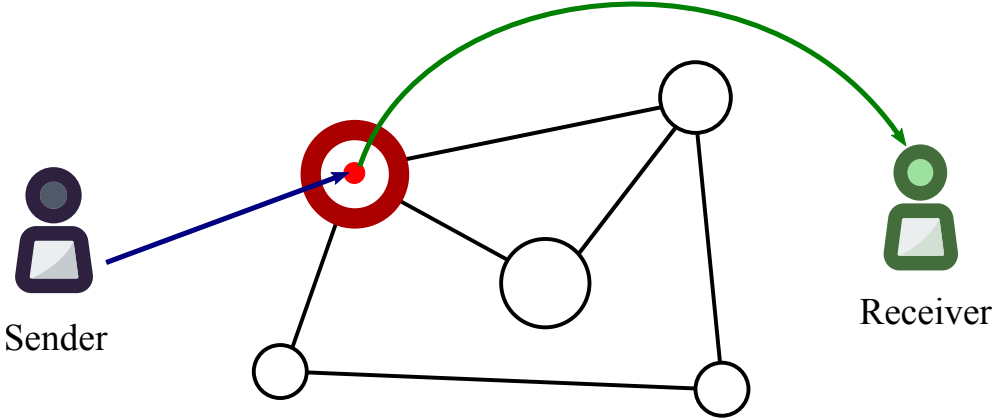


Decentralized Key Management using PRE

Using threshold split-key re-encryption (Umbral)

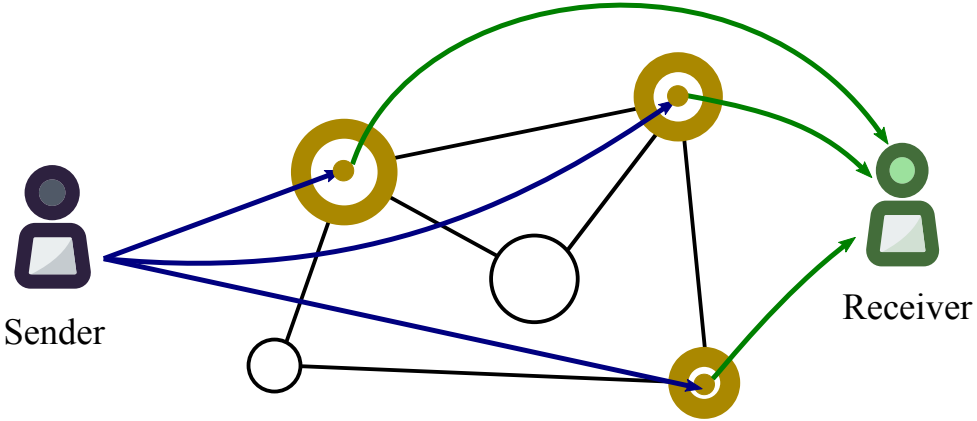


KMS Network: Data Sharing + PKE



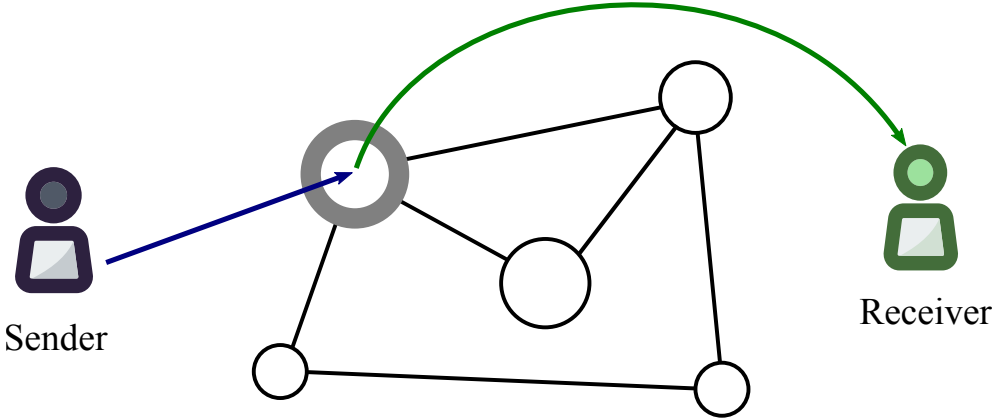
- Single node has access to data
- Single node can deny to do work

KMS Network: Data Sharing + PKE + Shamir Secret Sharing



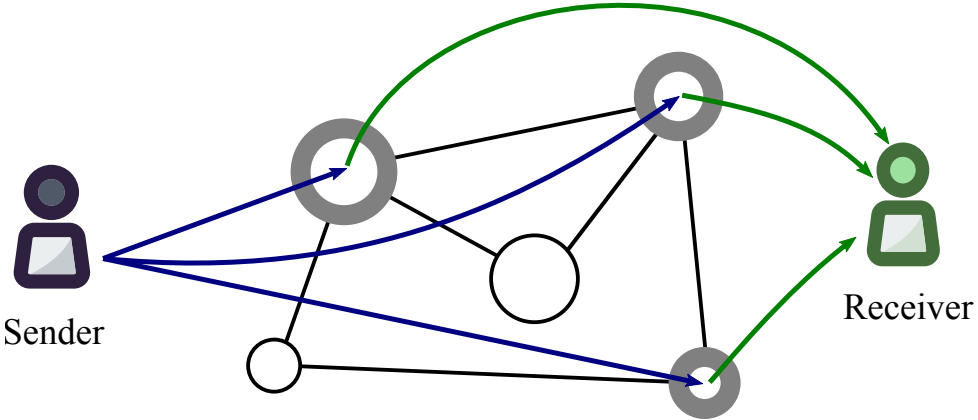
- Nodes can collude to gain access to data

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

NU Token

Purpose

- Splitting trust across re-encryption nodes
  - More tokens = more trust, more work, and more compensation
- Proof of Stake for minting new coins according to the mining schedule
- Security deposit at stake against malicious behavior of nodes

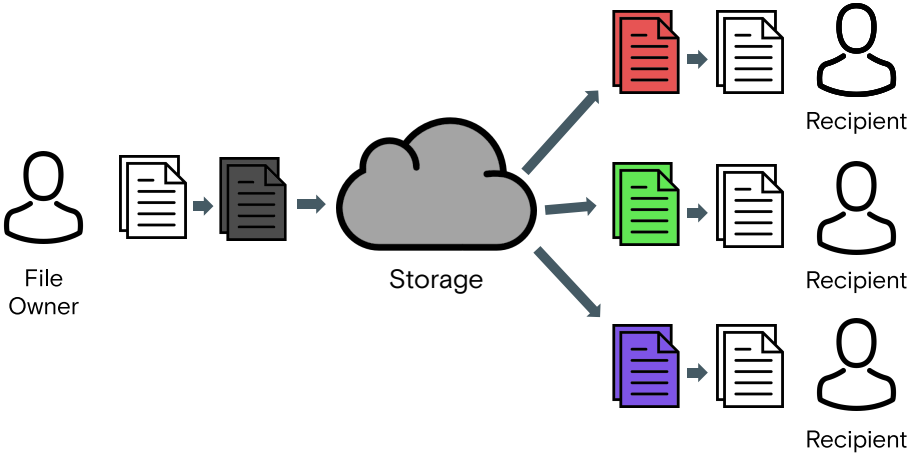
Data Sharing Policies

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

Decentralized re-encryption nodes (Ursulas) relied on to apply conditions without having the ability to decrypt data

Use Cases

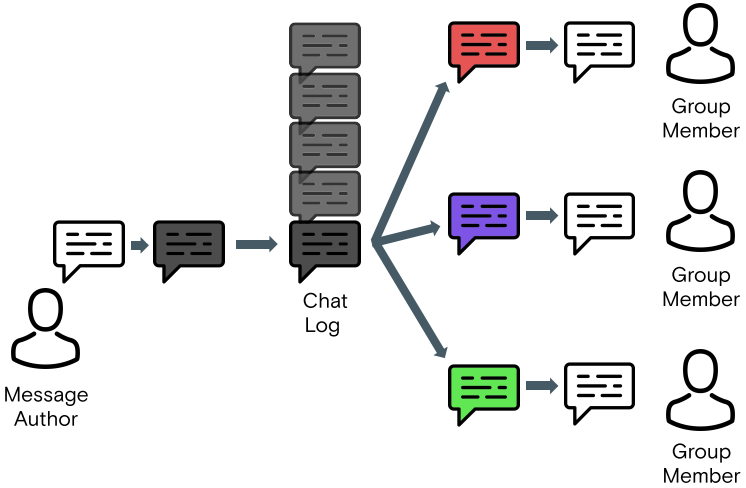
Encrypted file sharing



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

Use Cases

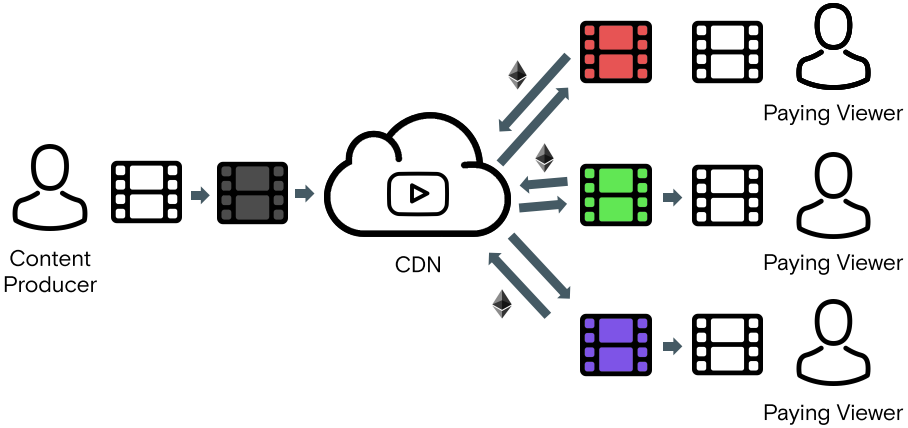
Encrypted multi-user chats



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

Use Cases

Decentralized access-controlled content



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

Decentralized Database

- bluzelle
- fluence.ai
- WOLK

Sharing Economy

- ORIGIN

Internet of Things

- SPHERITY
- XAIN.

User Data Monetization

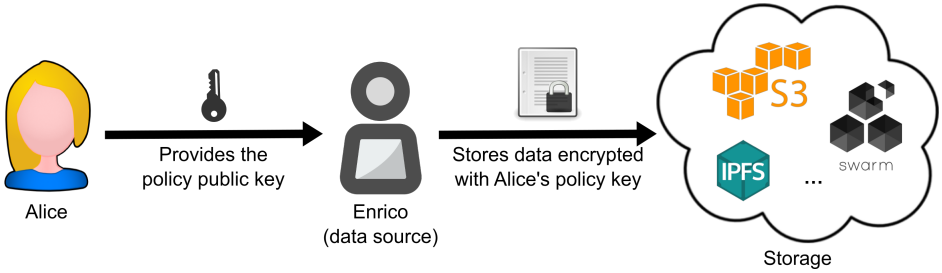
- SWIPE
- datum

NuCypher Characters

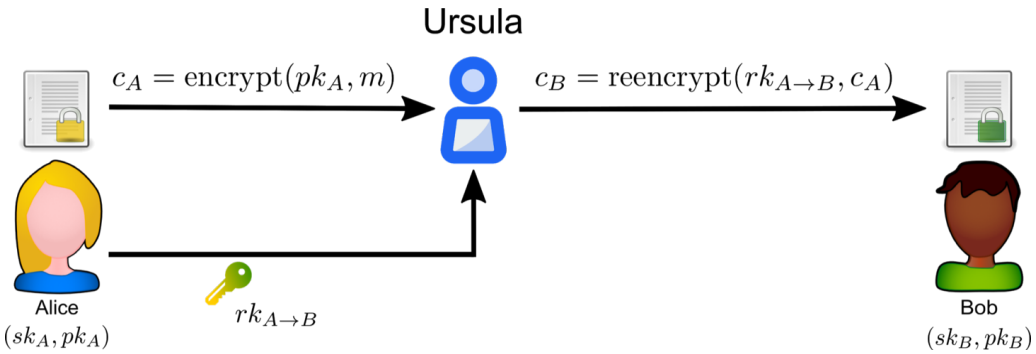
- Alice – data owner
- Bob – data recipient
- Enrico – data source
- Ursula – proxy node

Alice & Bob

Enrico

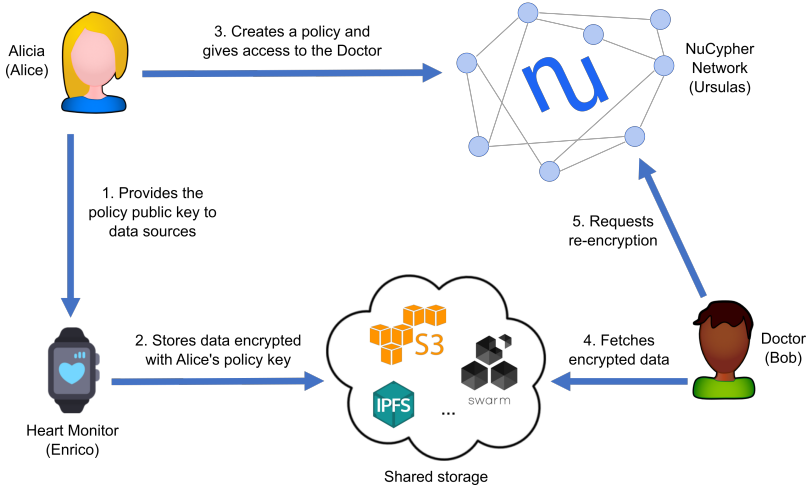

Alice
Provides the
policy public key
Enrico
(data source)
Stores data encrypted
with Alice's policy key
S3
IPFS
...
swarm
Storage

Ursula


Ursula
$c_A = \text{encrypt}(pk_A, m)$
$c_B = \text{reencrypt}(rk_{A \to B}, c_A)$
Alice
(sk_A, pk_A)
$rk_{A \to B}$
Bob
(sk_B, pk_B)

Heartbeat Demo

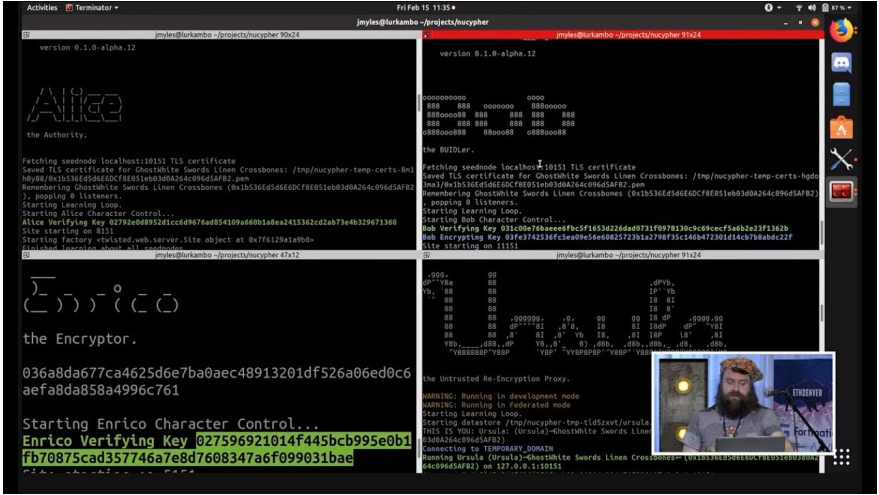
http://docs.nucypher.com/en/latest/demos/heartbeat_demo.html



Character Control Demo

Non-Python Usage

https://www.youtube.com/watch?v=om0tew-Z4gE



Competing Technology

Data Masking and Tokenization

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

Public Key Encryption

- Data must be decrypted before it is shared
- Not Scalable

Multi-Party Computation

- Interactive protocol
- Slow Performance

Fully Homomorphic Encryption

- Slow Peformance
  - NuCypher has developed a GPU-accelerated FHE library: nuFHE

Fully Homomorphic Encryption

nuFHE library

- GitHub: `https://nucypher.dev/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**	**-**

FHE Proof of Concept

Sputnik

- GitHub: `https://nucypher.dev/sputnik`
- Assembly language and interpreter for FHE that uses nuFHE
- Commits a merkle root of computation to the blockchain for proof of logic flow
- Used to execute first homomorphic smart contract at ETHBerlin 2018

ETHBerlin
@ETHBerlin

Follow

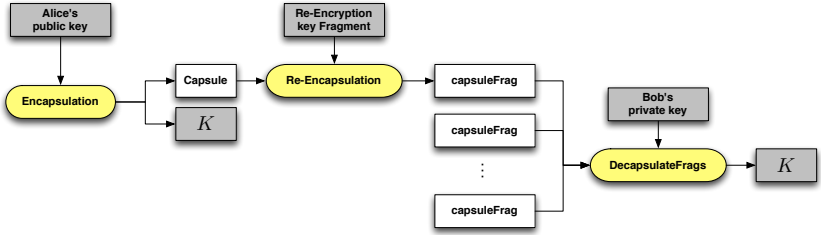
PLEASE give a round of applause to Sputnik!!! They are the first winners of our open track!! They designed A byte code assembly type language!YAAAAASSSS GUYS #ETHBerlin

More Information

NuCypher

Website: `https://www.nucypher.com`
Development Guide: `https://docs.nucypher.com`
Proxy Re-encryption Network: `https://nucypher.dev/nucypher`
nuFHE: `https://nucypher.dev/nufhe`
Hackathon Information: `https://nucypher.dev/hackathon`
Discord: `http://discord.nucypher.com`
E-mail: derek@nucypher.com

Appendix: Umbral Flow Diagram



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

Appendix: Umbral - Threshold Proxy Re-encryption

- "Umbral" is Spanish for "threshold"
- PRE properties: Unidirectional, single-hop, non-interactive
- Follows a KEM/DEM approach:
  - UmbralKEM provides the threshold re-encryption capability
  - Uses ECIES for key encapsulation with ZK proofs of correctness for verifiability on prime order curves (such as secp256k1)
  - 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: `https://github.com/nucypher/umbral-doc`

Appendix: Security Audits

Appendix: Fully Homomorphic Encryption