Building Zero Trust Computing Applications Using RAET

Samuel M. Smith Ph.D.
Founder
ProSapien
sam@prosapien.com

OpenWest 2017
Thursday July 13
Session 10:30 a.m. Room 300 C

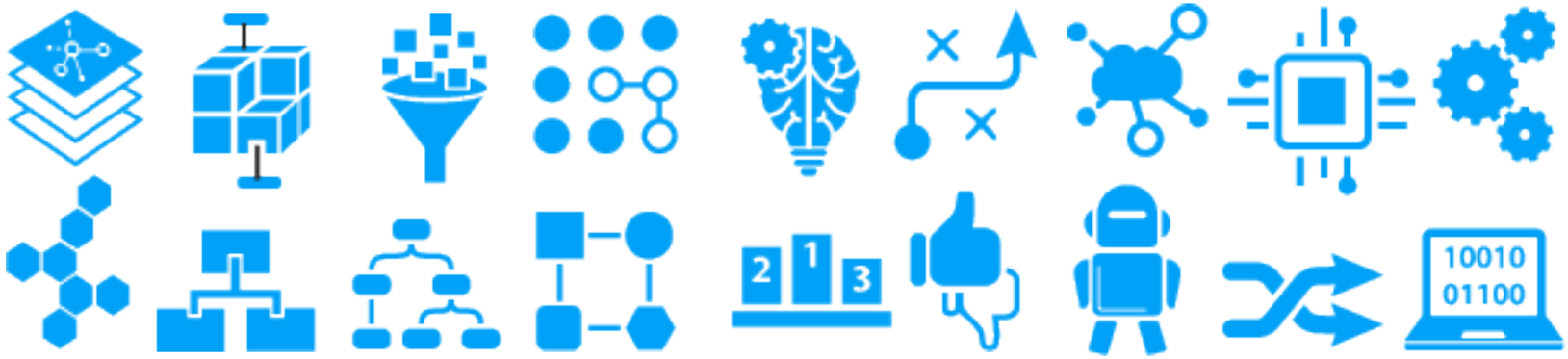
Shameless Plug

AI – Automated Reasoning

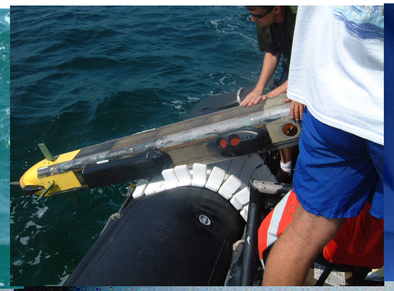
Distributed Consensus – Blockchain
Reputation – Identity

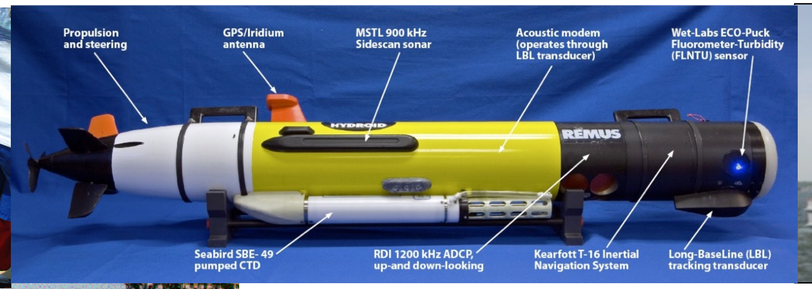
IIOT – Ephemeral Edge Computing

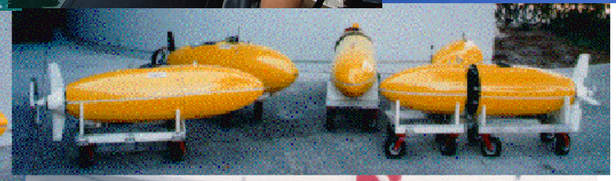
ProSapien

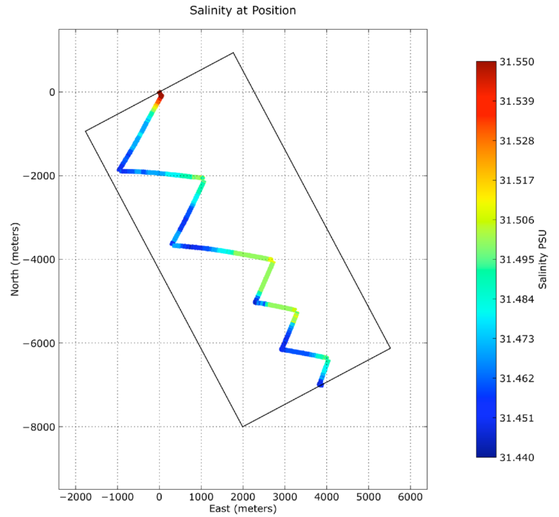


Autonomous Autonomic Automated









“Zero” Trust Computing?

No such thing as *zero* trust

Really its *diffuse* trust

zero trust term used in 2013NIST report

Diffuse trust perimeter-less security

Resources:

NIST: Developing a Framework to Improve Critical Infrastructure Cybersecurity 04/08/2013 Zero Trust Model for Information Security, Forrester Research.

http://csrc.nist.gov/cyberframework/rfi_comments/040813_forrester_research.pdf

https://www.nist.gov/cyberframework

Zero Trust Networks 2017 Gilman & Barth

https://www.amazon.com/Zero-Trust-Networks-Building-Untrusted/dp/1491962194/ref=sr_1_1?s=books&ie=UTF8&qid=1499871379&sr=1-1&keywords=zero+trust+networks

Diffuse Trust

Cellular distributed topology.

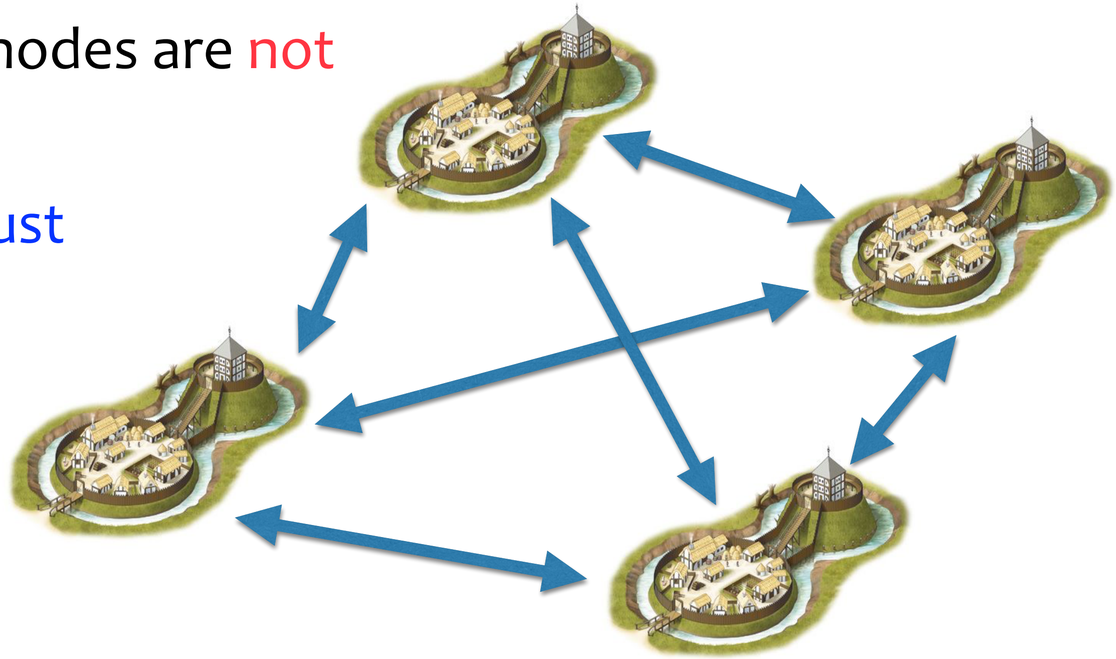
> *"Think clandestine spy ring or resistance organization"*

Defeating each node requires an independent exploit.

Caveats.

> No single point of failure – *"Universal root privileges"*
>
> No common mode failures – *"Exploit to attain root privileges"*

vs



The system is trustworthy as long as some majority of nodes are not exploited.

Distributed consensus is the key to achieving diffuse trust

Peer cooperation versus hierarchical control

Distributed Consensus

Enables diffuse trust systems using de-centralized computing infrastructure

Allows secure infrastructure outside a firewall by distributing the attack surface.

Enables the replacement of strongly controlled computing infrastructure with weakly controlled computing infrastructure that may be more secure.

Enables computation in the “*fog*” vs computation in the “*cloud*”.

Enables new disruptive business models

Distributed Consensus Types

Proof of Work: Simple, highly inefficient, high latency, low throughput

Proof of Stake: More complex, more efficient, lower latency, higher throughput (Asymmetric Proof of Work)

Byzantine Agreement: Most complex, most efficient, lowest latency, highest throughput. (Byzantine Fault Tolerant)

Hybrid: Side chains, anchoring,

Zero Trust Computing

Security, Privacy, Agency

Diffuse trust perimeter-less security model

Diffuse trust perimeter-less security principles

The network is always hostile, internally & externally; Locality is not trustworthy.

Every network interaction or data flow must be authenticated and authorized using best practice cryptography.

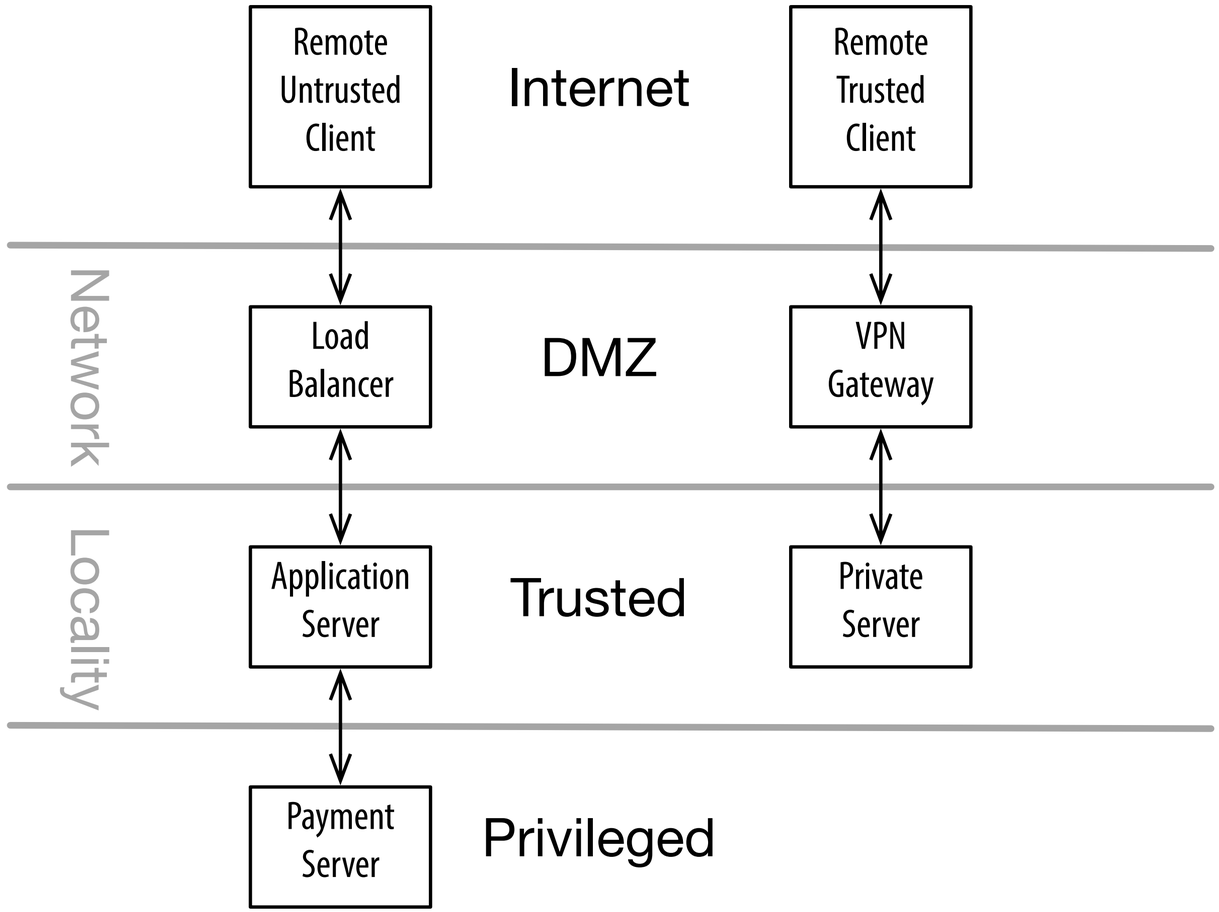
Inter-host communication must be end-to-end signed/encrypted and data must be stored signed/encrypted; Data is signed/encrypted in motion and at rest.

Policies for authentication and authorization must be dynamically modified based on behavior.

Policies must be governed by distributed consensus.

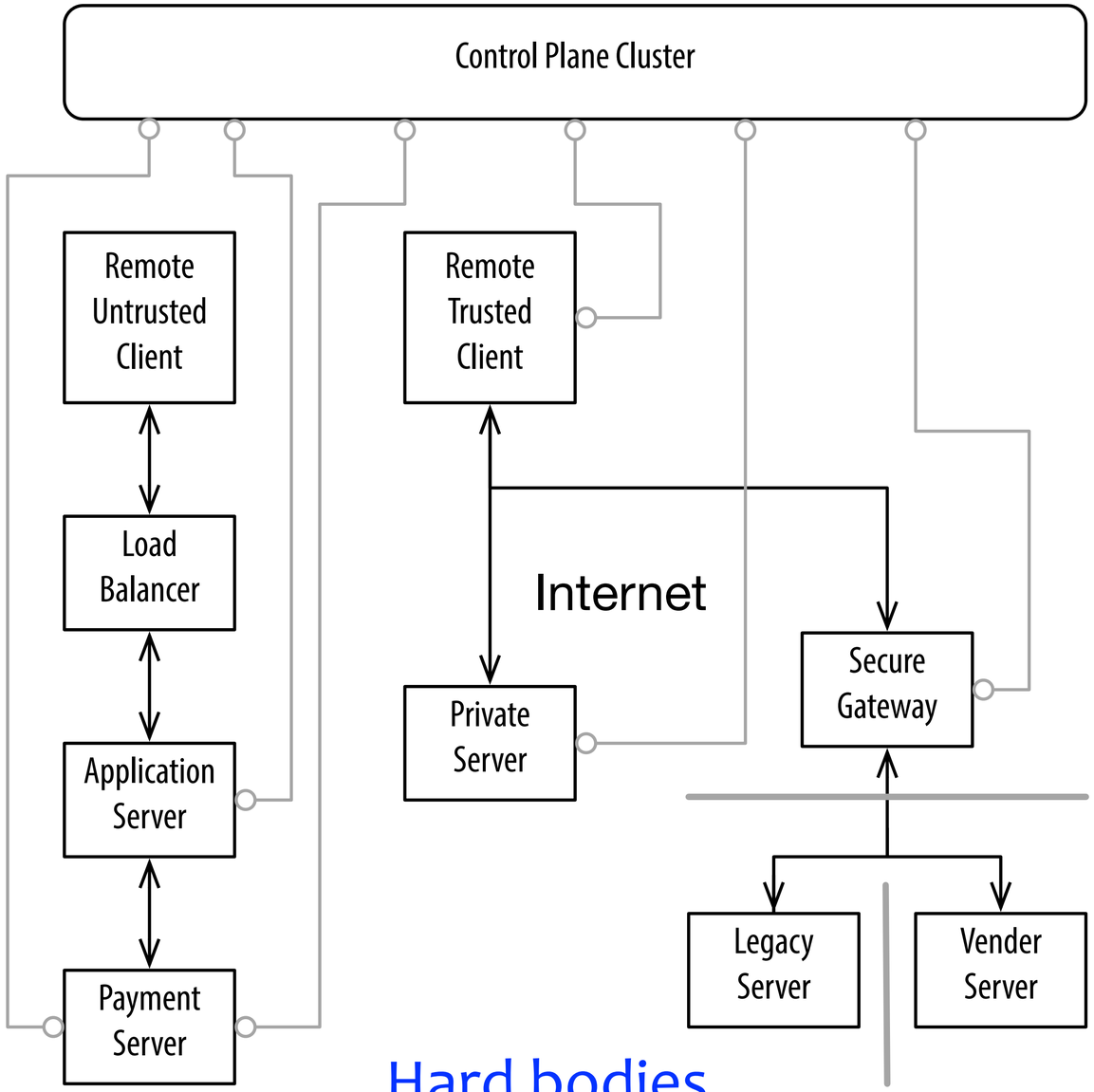
Security Models

Locality Trust



Hard shells – Soft bodies

Diffuse Trust



Hard bodies

DJB NaCL Cipher Suite

Best of breed Crypto Library is NaCl.

Designed by Daniel J. Bernstein, http://nacl.cr.yp.to

LibSodium: https://www.gitbook.com/book/jedisct1/libsodium/details

Python: LibNacl: https://github.com/saltstack/libnacl

Ed25519 (EdDSA) Signatures

Curve25519 (X25519) Diffie-Hellman Key Exchange

XSalsa20/20 Stream Cipher Encryption

poly1305 Message Authentication Code

CurveCP Protocol https://curvecp.org

NaCL: One True Cipher Suite

Ian Grigg: "*In the past, things like TLS, PGP, IPSec and others encouraged you to slice and dice the various algorithms as a sort of alphabet soup mix. Disaster. What we got for that favour was code bloat, insecurity at the edges, continual arguments as to what is good & bad, focus on numbers & acronyms, distraction from user security, entire projects that rate your skills in cryptoscrabble, committeeitus, upgrade nightmares, pontification ... Cryptoplumbing shouldn't be like eating spagetti soup with a toothpick. There should be One Cipher Suite and that should do for everyone, everytime. There should be no way for users to stuff things up by tweaking a dial they read about in some slashdot tweakabit article while on the train to work... Picking curve25519 xsalsa20 poly1305 is good enough for that One True CipherSuite motive alone... It's an innovation! Adopt it.*"

Matthew Green: "*Any potential 'up my sleeve' number should be looked at with derision and thoroughly examined (Schneier thinks that the suggested NIST ECC curves are probably compromised by NSA using 'up my sleeve' constants). This is why I think we all should embrace DJB's Curve25519.*"

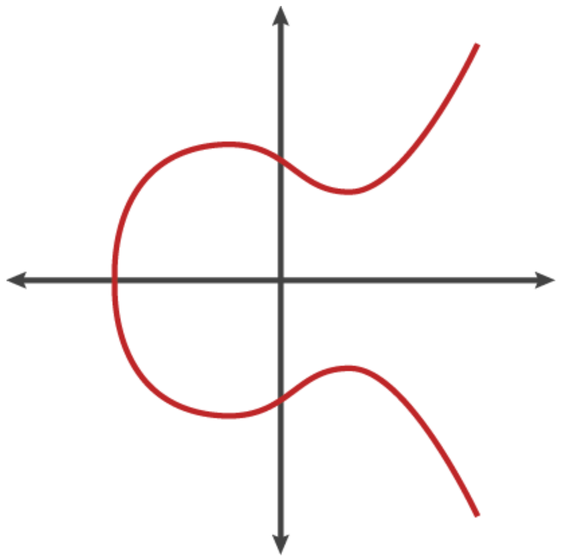
wolfSSL: "*Curve25519 so far is destroying the key agreement and generation benchmarks of previous curves, putting up numbers for both key agreement and generation that are on average 86 percent faster than those of NIST curves."*

Adam Langley: "*Of the concrete implementations of Diffie-Hellman, curve25519 is the fastest, common one.*"

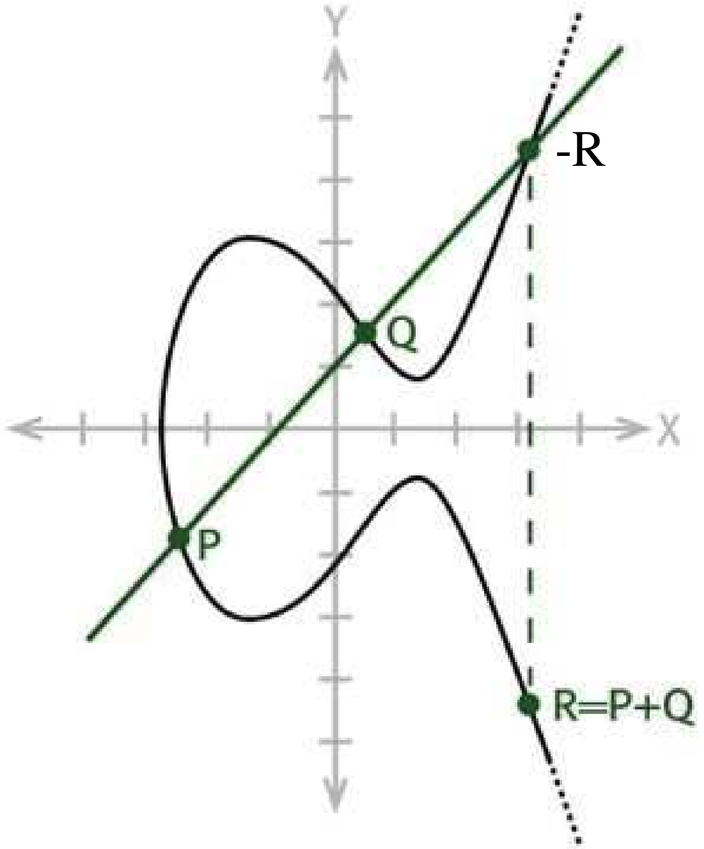
Dan Bernstein: "*An attacker who spends a billion dollars on special-purpose chips to attack Curve25519, using the best attacks available today, has about 1 chance in 1,000,000,000,000,000,000,000,000,000 (27 0s) of breaking Curve25519 after a year of computation.*"

Modern Crypto: Elliptic Curves

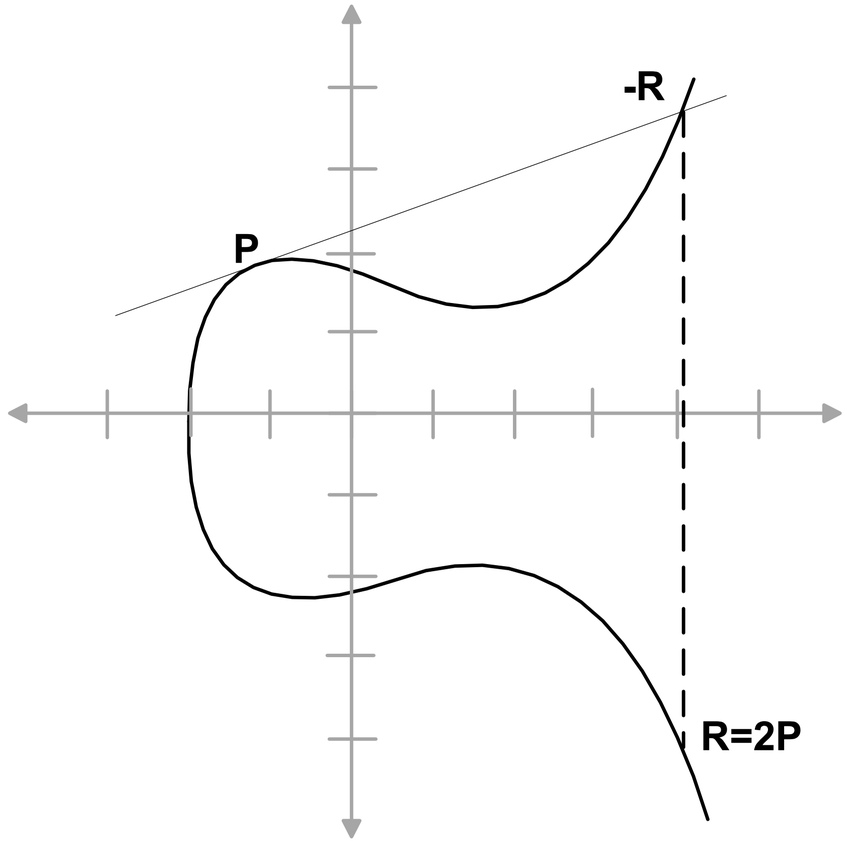
$y^2 = x^3 + ax + b \mod p$



$R = P + Q$



$R = P + P = 2P$



Modern Crypto: ECC Diffie-Hellman Key Exchange

Given elliptic curve polynomial coefficients *a*, *b*, mod prime *p* and, generator point *G* on curve.

(*Public, private*) key pair *(K, k)* where *K =k*G*,
k is random number,
* means ECC scalar multiplication by adding *G*, *k* times
scalar multiplication is commutative
K is another point on the curve

Alice picks private key *a*, random number, and generates public key *A*, point on curve. A = a*G

Bob pick private key *b*, random number, and generates public key *B*, point on curve. B = b*G

Alice and Bob exchange public keys *A* and *B*.

Alice generates shared secret key *S*, point on curve, from Alice's private key *a* and Bob's public key *B*.

S = a * B = a * b * G

Bob generates shared secret key *S*, point on curve, from Bob's private key *b* and Alice's public key *A*.

*S = b * A = b * a * G*

*S = b*a*G =a*b*G*

Now S can be used as source of shared secret to encrypt/ decrypt messages between Alice and Bob.

RAET Reliable Asynchronous Event Transport

https://github.com/RaetProtocol/raet

End-to-End Encryption and Signing with:

NaCL, X22519, CurveCP Handshake, Curve22519, Salsa20/20 Poly1305

Python with LibNacl, Ioflo

UDP for interhost communication

Unix domain sockets for interprocess communications

Presence Support

Truly asynchronous architecture with non-blocking I/O

Peer-to-peer bidirectional Curve-CP handshake

Curve-CP Handshake

Confidentiality against espionage

Integrity against corruption and forgery

Some availability against sabotage

Server authentication

Client authentication

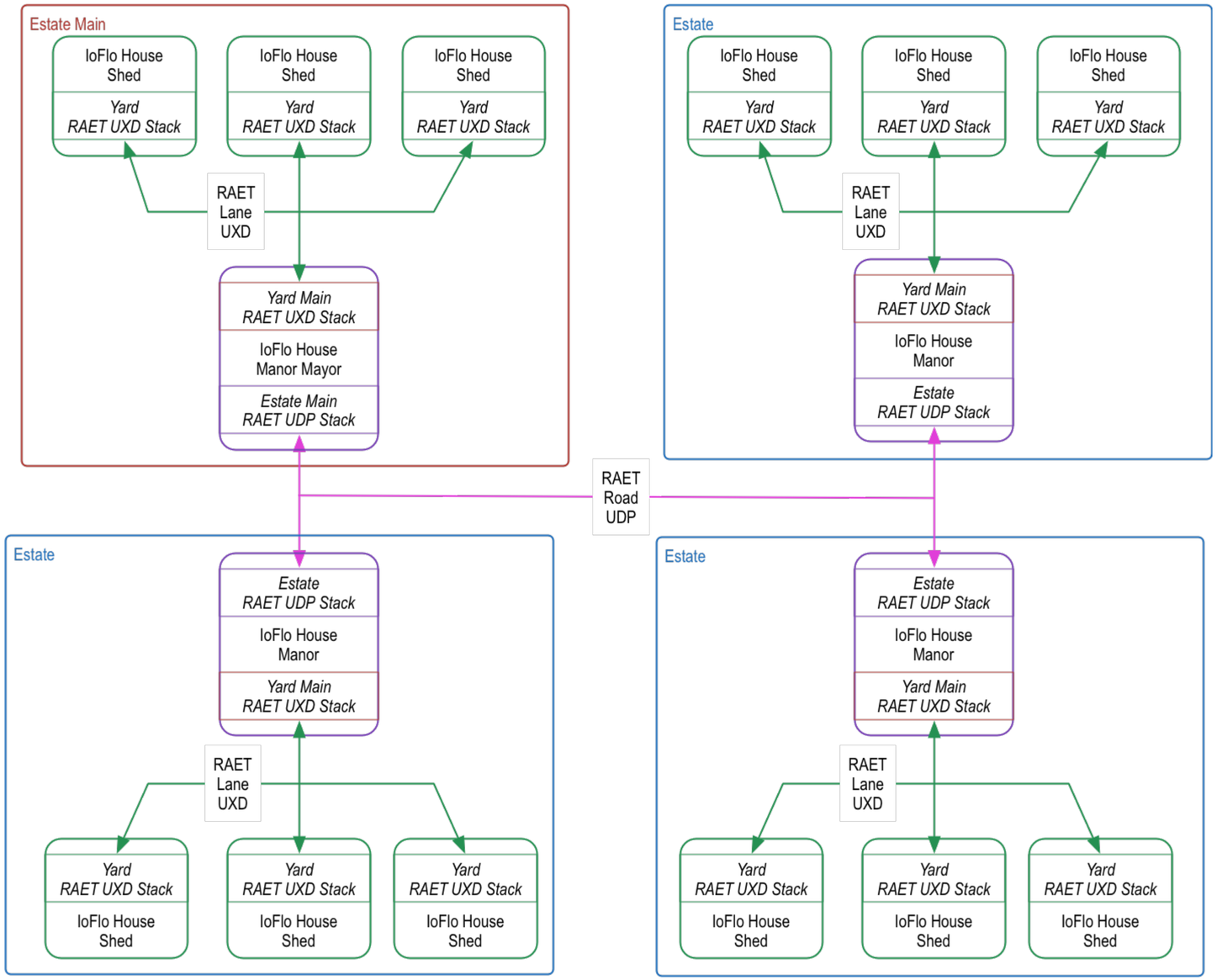
Replay attack protection

Man-in-the-middle attack protection

Passive forward secrecy

Active forward secrecy

RAET Metaphor



RAET Transactions

Join: Exchange long term public keys both signing and encryption.

Configurable to be compatible with multi-factor authentication

Allow: Curve-CP handshake

exchange short term ephemeral encryption keys (diffie-hellman)

Message: Exchange messages

Header is compact text, JSON, or Msgpack

Body is raw, JSON, or Msgpack

Max message size up to 67,107,840 bytes

Message bodies are encrypted

Heady + Body are signed

Alive: Presence heartbeat

RAET Key & Channel Management

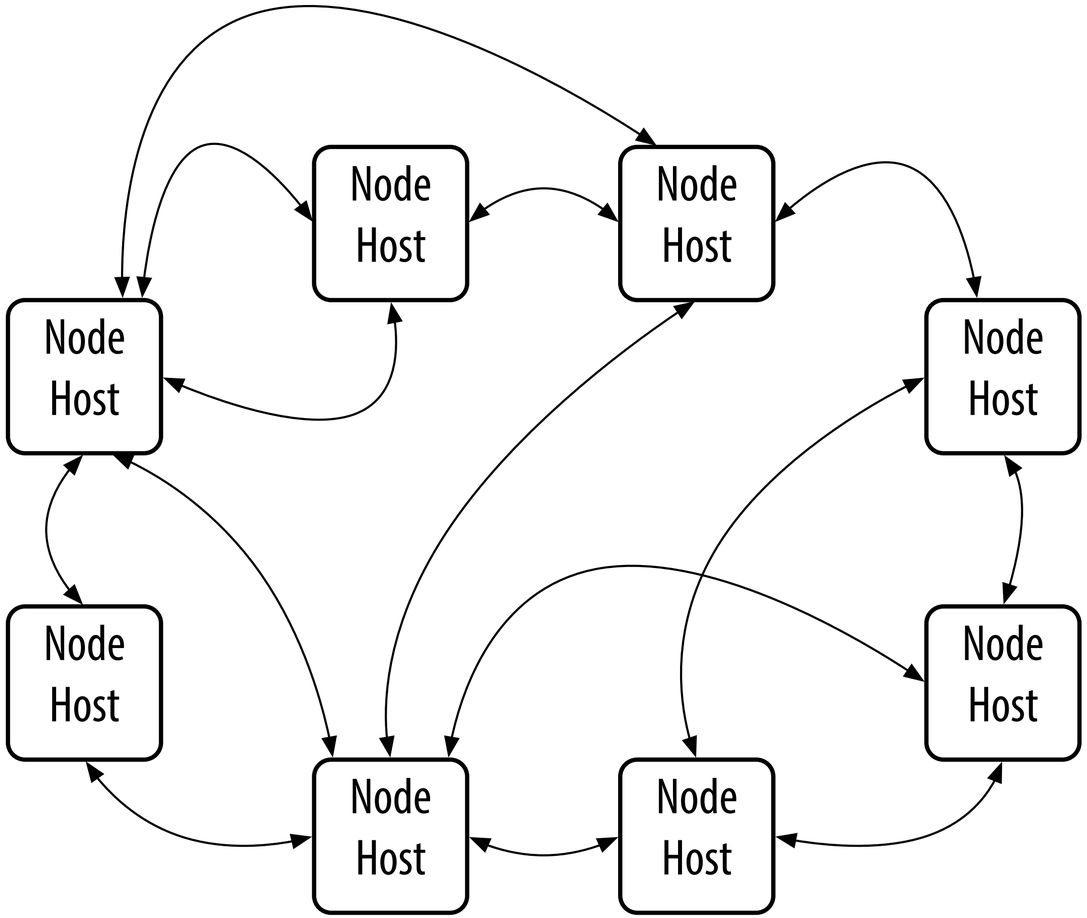
Pend, Accept, Reject

Name, Role, Mode

Mutability

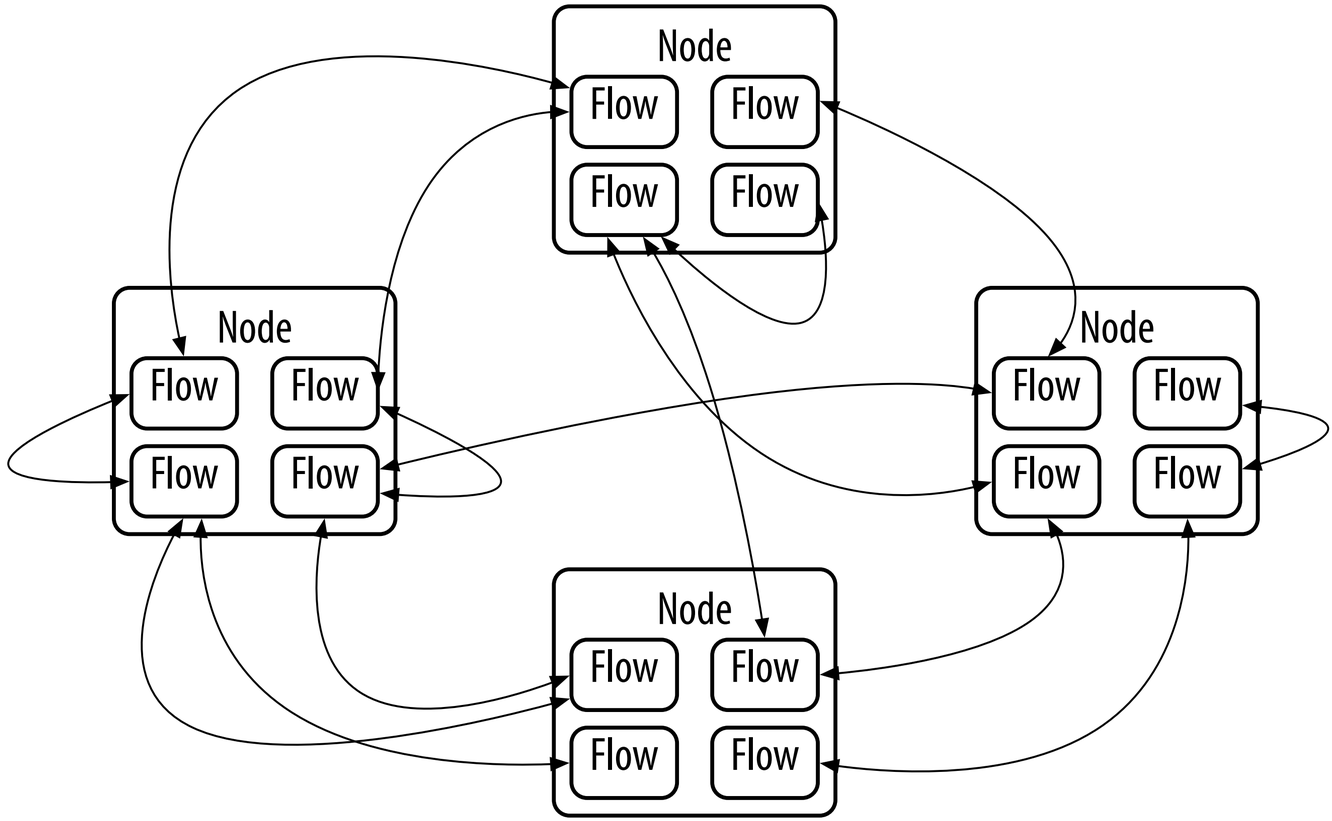
Many-to-Many-to-Many

Many-to-many



Performance and Reliability

Many-to-many-to-many



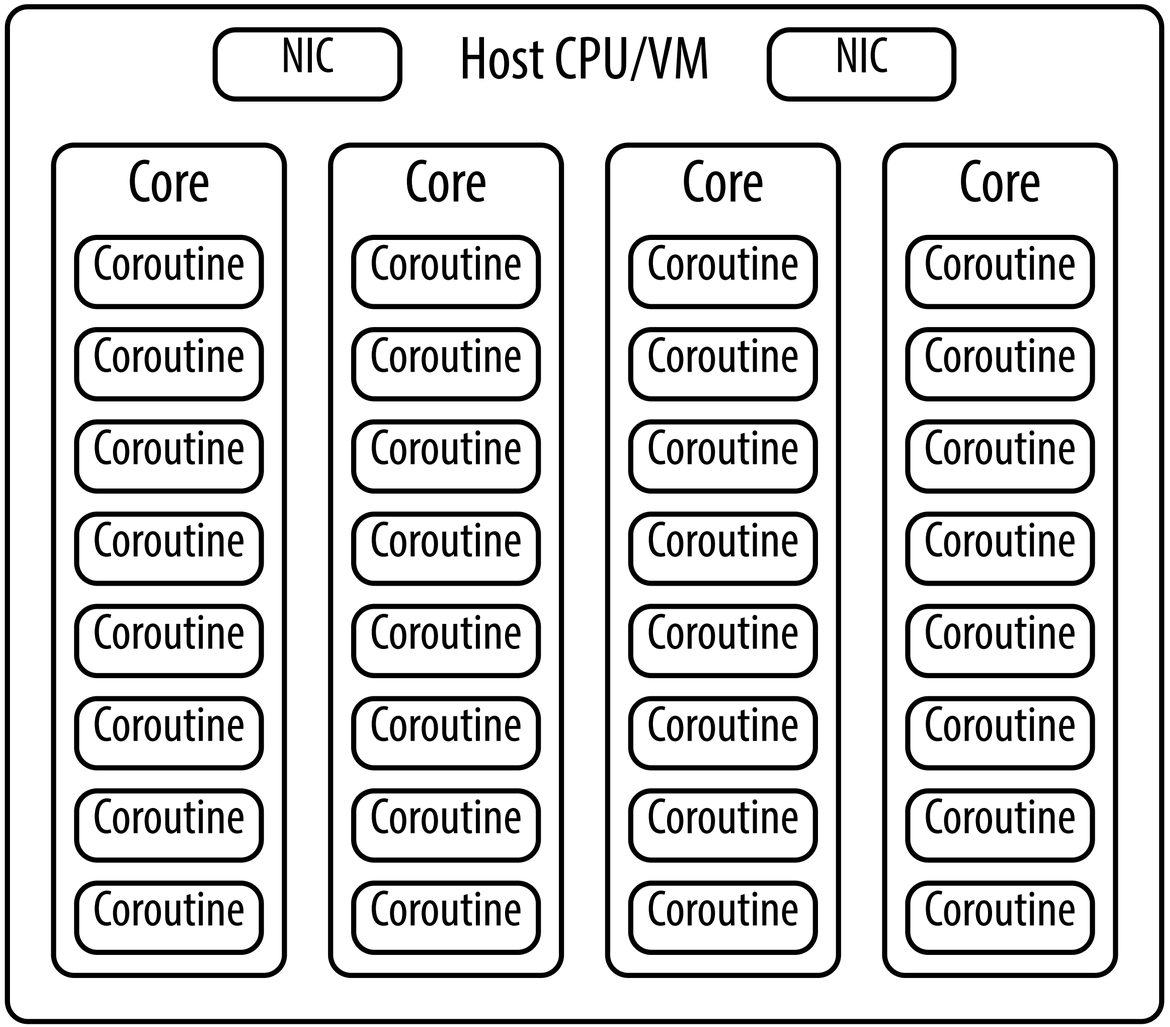
Logically concurrent flows of execution

Concurrent data flows

Service flows

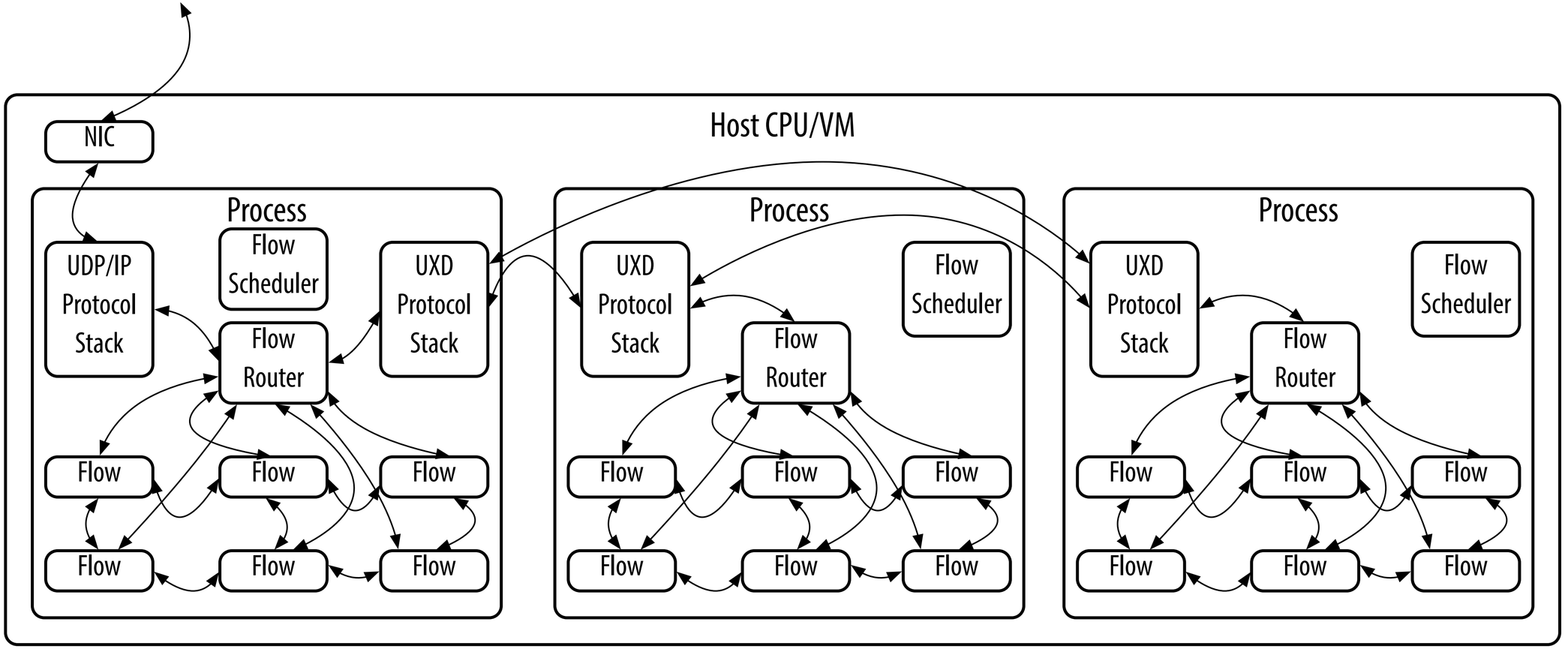
Asynchronous Optimized Computation

Coroutines are the fundamental units of logically concurrent computation



Minimizes context switch overhead

Data Flow Routing



Enables coroutine-to-coroutine communication
both intra-process, inter-process and inter-host

Encapsulated Routing Data

Tuple: (datashare name, UXD name, UDP name, channel name)

JSON:

```
{
    "route":
    {
        "src": ["peter.piper.picked", "pepper", "peck", "garden"],
        "dst": ["suzy.sand.sell", "shell", "shore", "sea"]
    }
    "tag": "commerce.hot",
    "stamp": "20170125T13:45:62.340123UTC",
    "data":
    {
       "quantity": 5,
       "price": 1.50
    }
}
```

RAET Features

https://github.com/RaetProtocol/raet

Peer-to-peer exchange of long term keys with multiple levels of control over how the exchange occurs.

Peer-to-peer implementation of the CurveCP handshake that allows for either side of a pair of hosts to initiate the handshake and resolves simultaneous CurveCP handshakes.

Exchanges name, role, and mode identifiers. Completed handshake authenticates the name for data flow routing, the role for authorization, and the mode for how the node operates on a given channel.

Extensible protocol that makes a trade-off on the side of simplicity and ease of debugging with a flexible header and a flexible data body.

The header by default is ascii readable and achieves compactness using a default value policy. This makes the ascii header on average about the same size as a full binary header.

Because the header is not fixed it is easily extensible and future proof.

Using routed data flows on top of the RAET channel credentials enables a data flow router to easily implement authorization policy. Where Authorization is per data flow address. Dynamic authorization policy can be easily implemented via a lookup table associated with each router.

Hyperledger Plenum RBFT Python Implementation

Plenum RBFT https://github.com/hyperledger/indy-plenum

https://github.com/evernym/plenum/wiki

Digital Signatures (no Macs) so don't need digests.

Uses RAET (Reliable Asynchronous Event Transport) protocol

https://github.com/RaetProtocol/raet

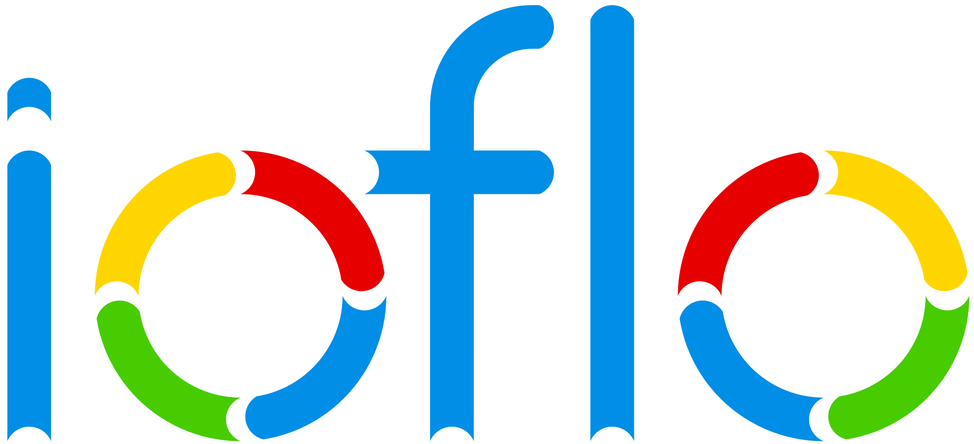
Projects Related to RAET

Plenum RBFT https://github.com/hyperledger/indy-plenum

https://sovrin.org

https://www.hyperledger.org

https://github.com/saltstack/raet



https://github.com/ioflo/ioflo

Running RAET Asynchronously

https://github.com/RaetProtocol/raetasync

Sleeping loop

Gevent Greenlets

Generators

Coroutines

Ioflo

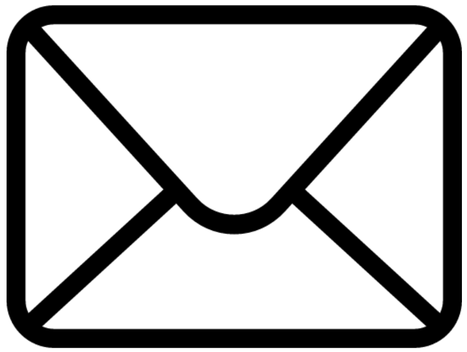
Background Slides

Modern Crypto: Zero Knowledge

Trusted first party Prover, proves to second party, Verifier that a statement is true in such a way that secret information upon which the proof of the statement is based is not conveyed to the Verifier. The Verifier cannot then prove to a third party that the statement is true.

Modern Crypto: Blinded Signature

Party A writes message destined for C.
Party B verifies A's identity.
A writes message on paper but does not disclose message to B.
A puts message in carbon paper envelope.
B signs outside of envelope and transmits envelope to party C.
Party C opens envelope and knows message came from a verified identity by the carboned signature of B
C cannot trace the contents of the message back to A.
B cannot disclose the message to anyone besides C, because B never saw the message.



Modern Crypto: Secret Sharing

Voldemort Horcrux.

Split secret into pieces and distribute them.

N of M pieces needed to reconstruct secret.

Key Recovery.

Data Storage.

Distributed Autonomic Service

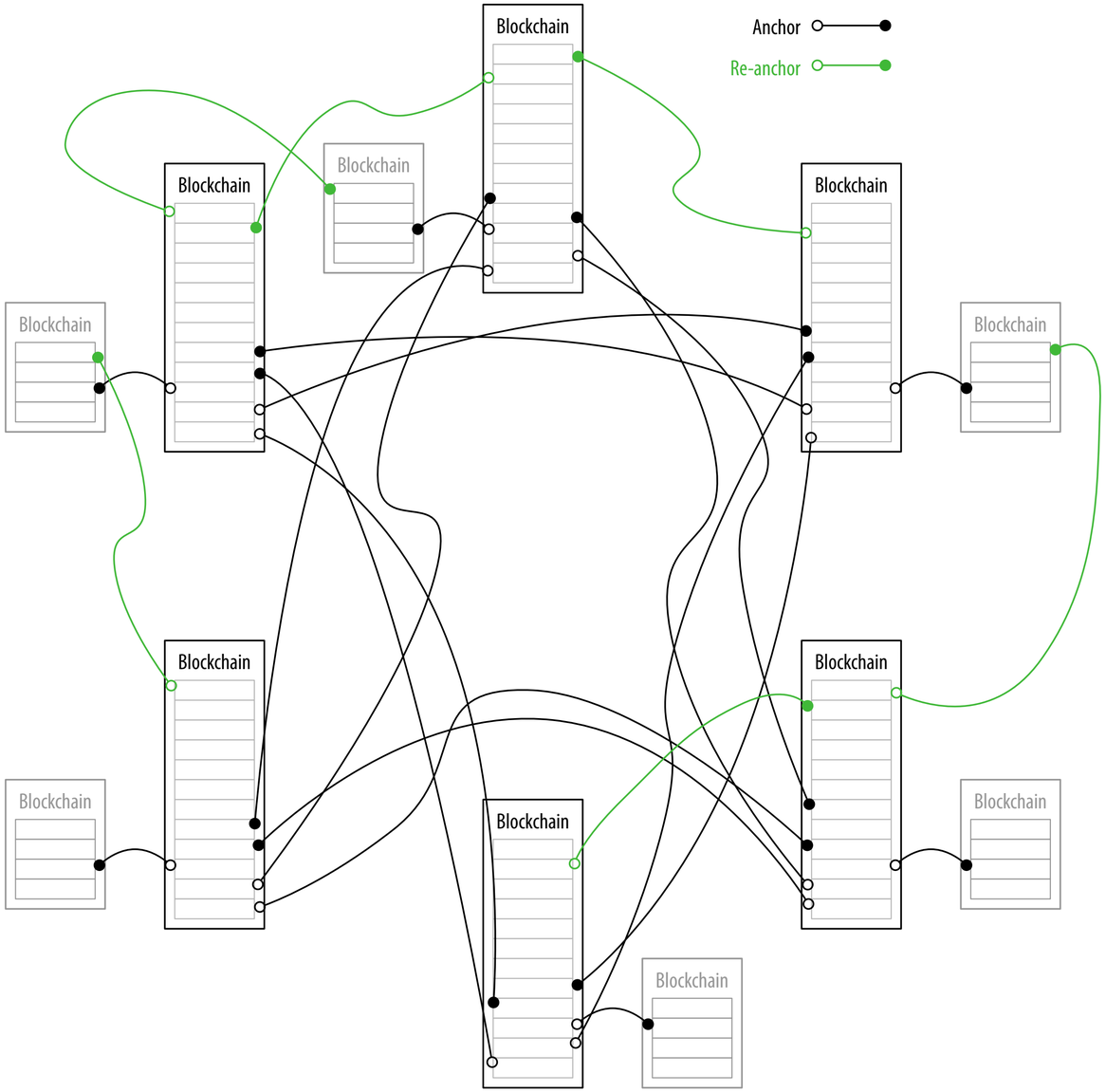
http://xaltry.com

DAS = service based on distributed consensus + autonomic computing algorithms on decentralized computing infrastructure = *distributed AI*

Xaltry *reputation as a service* (RAAS)

Eventually a RAAS on a DAS

Chainmail or Mesh Anchoring



What is BlockChain Technology?

Euphemism for a host of related technologies and concepts of which blockchains are not the most important

Byzantine Generals

- Analogy to war in the Byzantine era where generals were prone to duplicitous acts
- The Byzantine Generals Problem:

  http://research.microsoft.com/en-us/um/people/lamport/pubs/byz.pdf

  Separated generals agree to attack a city. If they unite they win. If not the attacker(s) lose.

  The communication of intent to attack and confirmation of intent to attack is subject to delay and error.

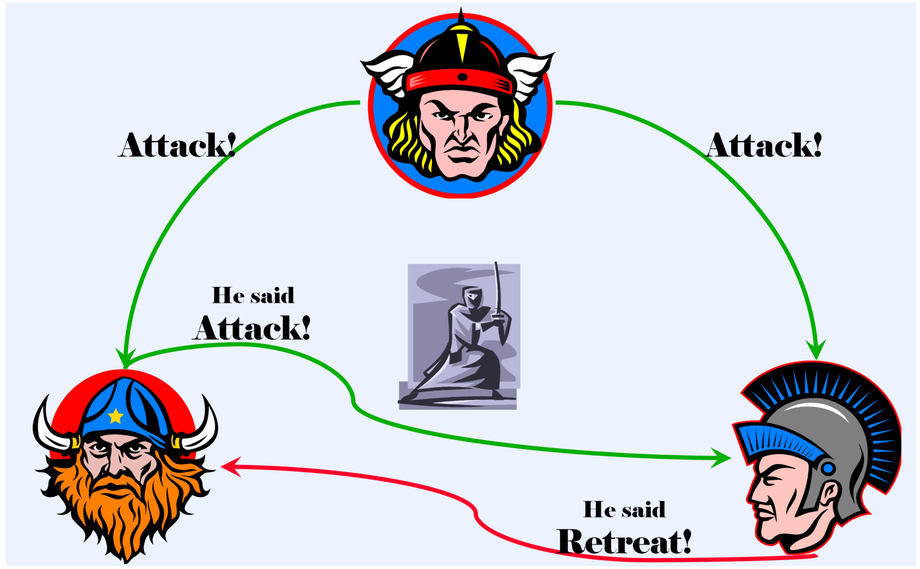
- Byzantine agreement:

  A valid consensus is guaranteed despite a fraction of the participants behaving in a malicious and/or erroneous manner.

- Consensus despite “Byzantine” faults:

  Dropped, erroneous, or malicious Packets. Partitioning. Sybil attacks.

  A consensus algorithm that achieves agreement despite the presence of Byzantine faults is called Byzantine fault tolerant (BFT).



Graph Based Identity

Identity = *Identifiers* + *Attributes*

Identifiers = *globally unique* *cryptonyms* + *aliases*

Attributes = *user data, proofs*

Facilitate attribute exchange between entities sufficient to enable transaction to proceed

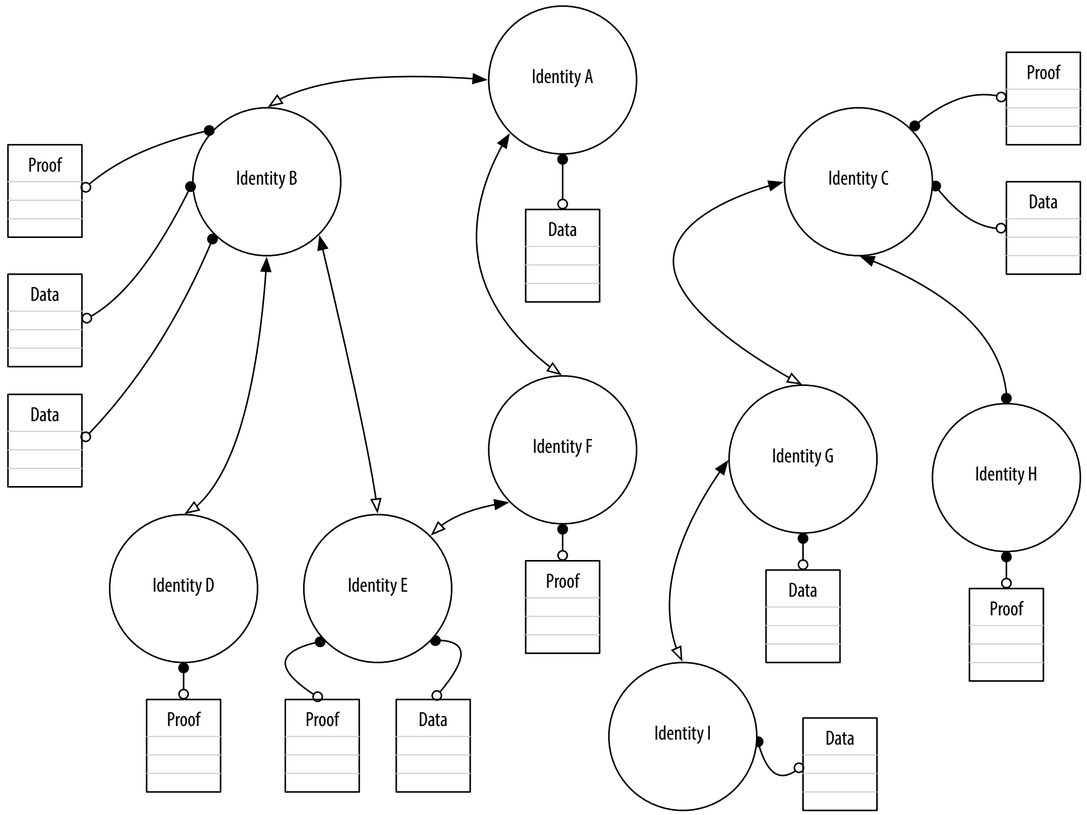
Identity System Features:

- Sovereignty (own your own identity)
- Security (impervious to fraud)
- Privacy (least disclosure)

Sovereignty = portable identifiers + user controlled access

Security = distributed consensus + modern crypto

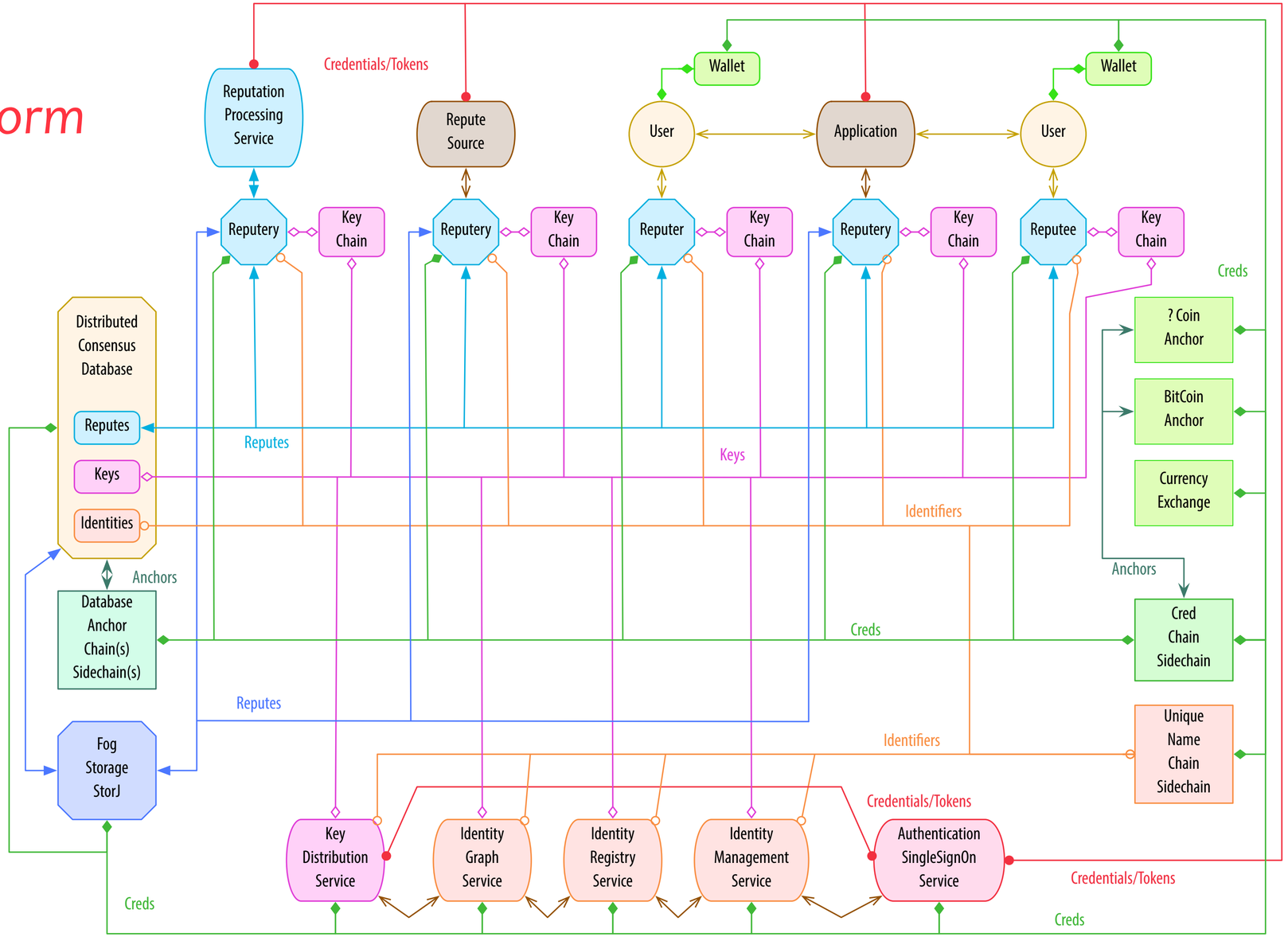
Privacy = granular graph based identities + layered disclosures + zero knowledge disclosures + group identities



https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/Identity-System-Essentials.pdf

Reputation System

Key component of any *platform*



https://github.com/SmithSamuelM/Papers/blob/master/whitepapers/open-reputation-low-level-whitepaper.pdf

Simple Reputation