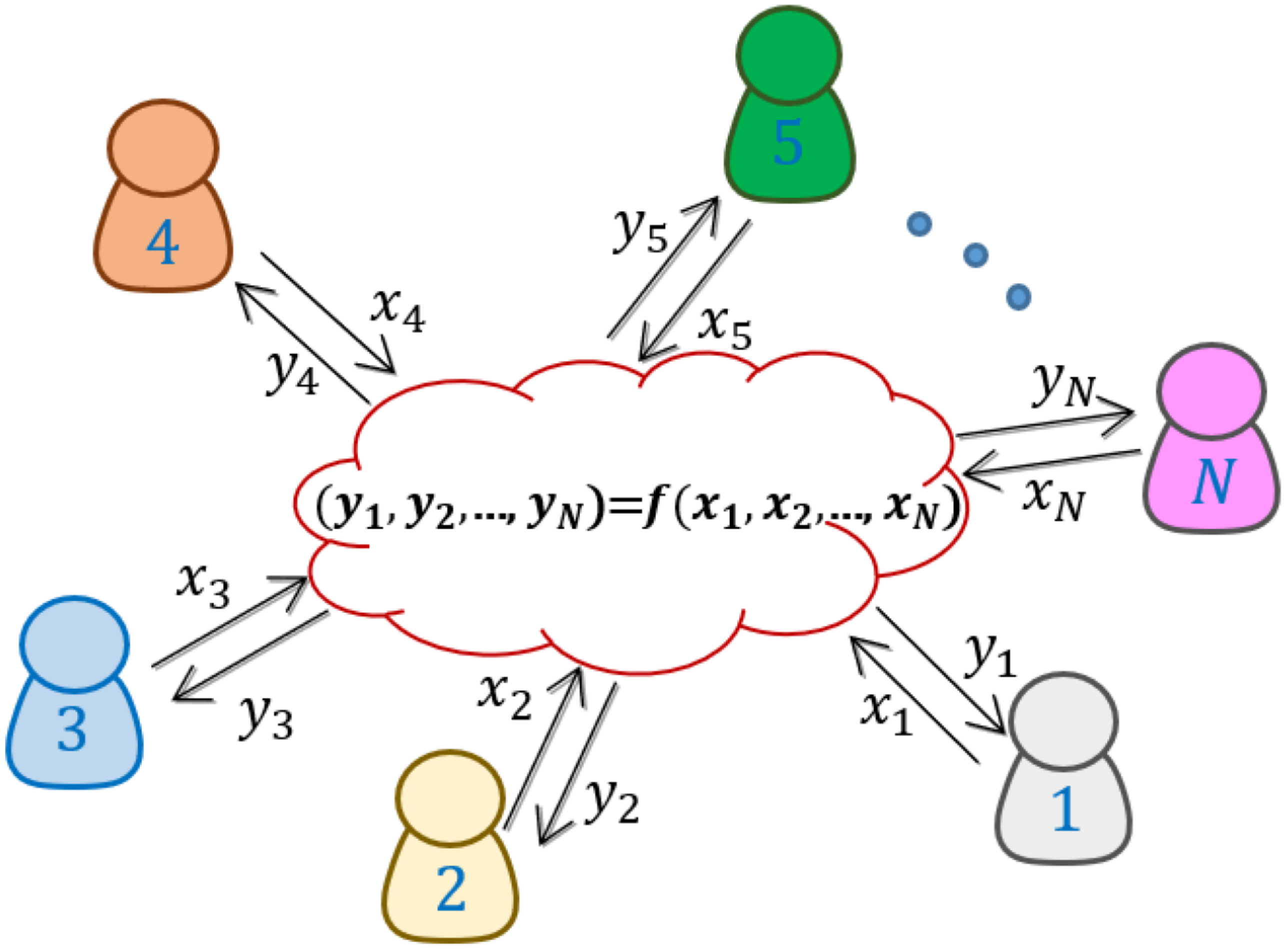
ECE 209AS (2/18/2020) Mid Term Presentation

P2I: Privacy Preserving Inferencing for Medical Cyberphysical Systems

Team Members: **Brian Wang, Swapnil Sayan Saha and Vivek Jain**

Presenter: **Swapnil Sayan Saha**

Motivation



How to **perform collaborative computation** in medical CPS **without leaking privacy sensitive data to competitors?**

SECURE MULTIPARTY COMPUTATION (SMPC)

SECURE AGGREGATION (SA)

Scalability, computational and implementation issues for both solutions with notion of trusted third parties (TTP)

Overall Project Goals and Specific Aims

Goals

- Implement SA and SMPC on the edge in MCPS
- Benchmark performance, privacy and security metrics of protocols
- Attempt to reduce computational overhead in SA and SMPC
- Erase the notion of TTP

Deliverables

Benchmark and tune SMPC and SA protocols for resource-constrained settings

↓

A real-time scalable and robust privacy preserving inferencing system at the edge for MCPS

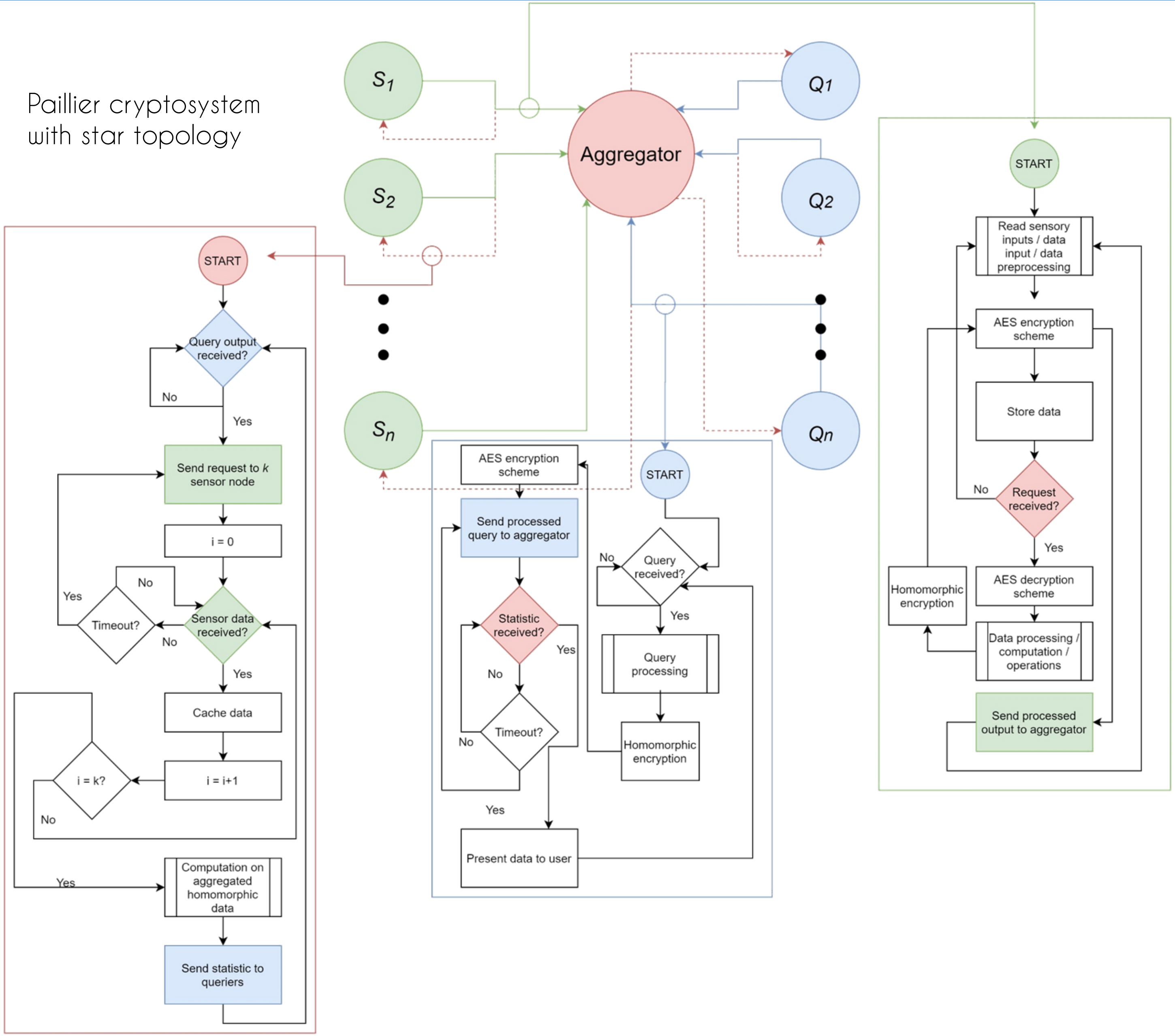
Attack Model

- **Secure Aggregation:**
  - Consists of central aggregator and adversary.
  - Aggregator: Semi-honest (cannot delete or ignore data/queries but curious).
  - Adversary:
    - Semi-honest (curious) (current setup).
    - Malicious (may provide malicious queries or data and curious) (future setup).
  - Adversary can be any of the parties excluding aggregator.

- **SMPC:**
  - No central entity (distributed system).
  - Computation is distributed among k nodes.
  - Adversary:
    - Semi-honest (current setup)
    - Malicious (future setup)
  - Zero-knowledge proofs for authentication of nodes and statistic.

Technical Approach (Secure Agg.)

Paillier cryptosystem
with star topology



Key Generation:

- $p, q \in P$ with equal length
- $n = pq$
- $g = 1 + n$
- $\phi(n) = (p-1) \cdot (q-1)$
- $\mu = \phi(n)^{-1} \bmod n$ (μ is used as *private-key*)

Encryption:

- Plaintext $m < n \ (m \in Z_n)$
- Choose $r < n$ dengan $\gcd(r, n) = 1$ randomly ($r \in Z_n^*$)
- Ciphertext $c = g^m . r^n \bmod n^2$

Decryption:

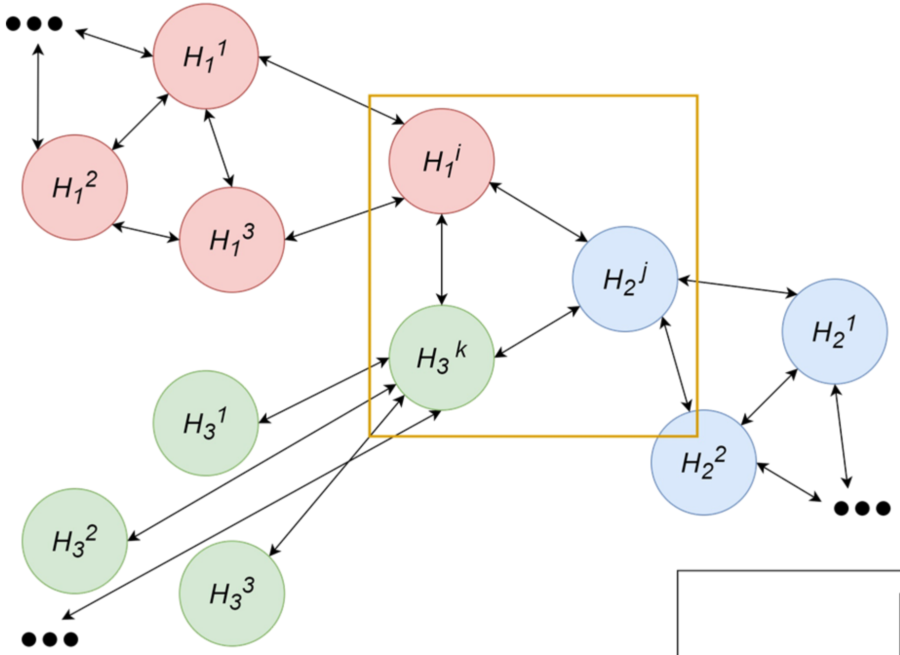
- Ciphertext $c < n^2$
- Plaintext $m = L(c^\lambda \bmod n^2) \cdot \mu \bmod n$

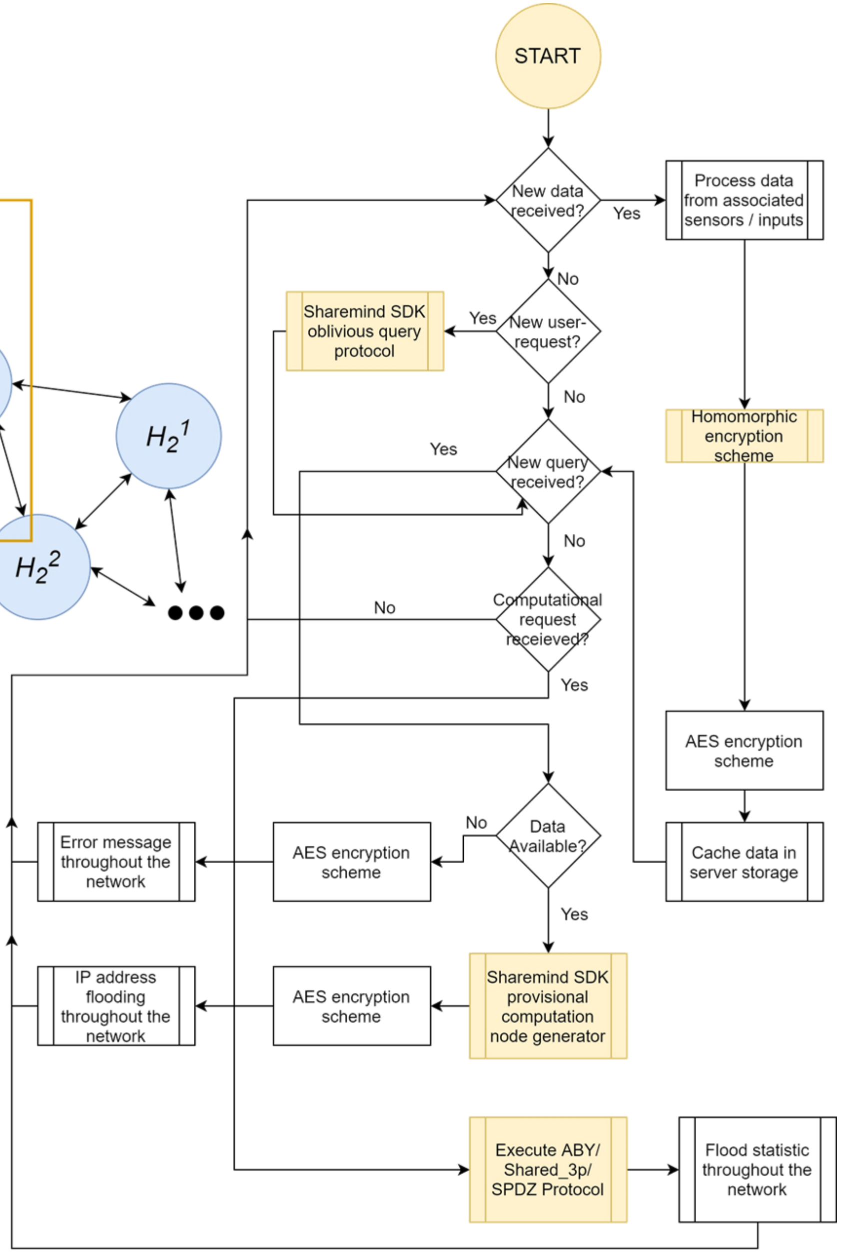
Technical Approach (Secure Agg.) (cont.)

- Operations implemented (Language: Python):
  - Mean
  - Convolution
  - Linear Regression
  - Vector Sum

- Preliminary benchmark (Intel I7 6700HQ – 16 GB RAM):
  - CPU usage of key generation: 17.6% on Intel I7 6700HQ
  - RAM consumption of key generation: 8MB
  - Time taken for key generation: 259.37ms (averaged using 100 key generations)
  - Time taken for encryption: 13.75 ms
  - Time taken for decryption: 15.625 ms
  - Time Elapsed for scalar addition: 46.875000 ns
  - Time Elapsed for scalar multiplication: 109.375000 ns

Technical Approach (SMPC)





sharemind

CYBERNETICA

Technical Approach (SMPC) (cont.)

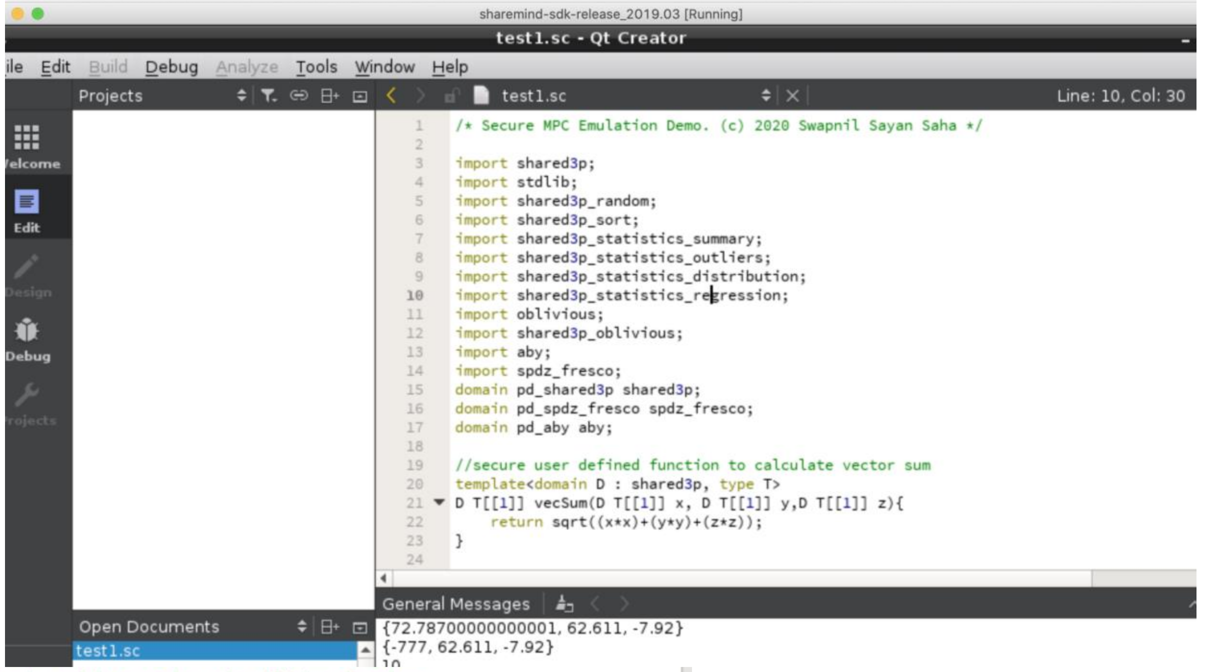
Implemented SMPC operations (Language: SecreC)

Shuffle	Quicksort	Outer join
Union	Intersection	MAD
Mean	Median	Upper Quantile
Lower Quantile	Minimum	Maximum
StdDev	Variance	Vector Sum (VS)
Outlier_MAD	Outlier_Quantile	Linear Regression
Obv_Insert		

Preliminary benchmark results* (-microseconds (CPU usage)):

	ABY	Shared 3p	SPDZ Fresco
Scalar Addition (+ encryption)	11373 (9%)	104 (11%)	18149 (10%)
Scalar Multiplication (+ encryption)	8055 (11%)	397 (14%)	25685 (10%)

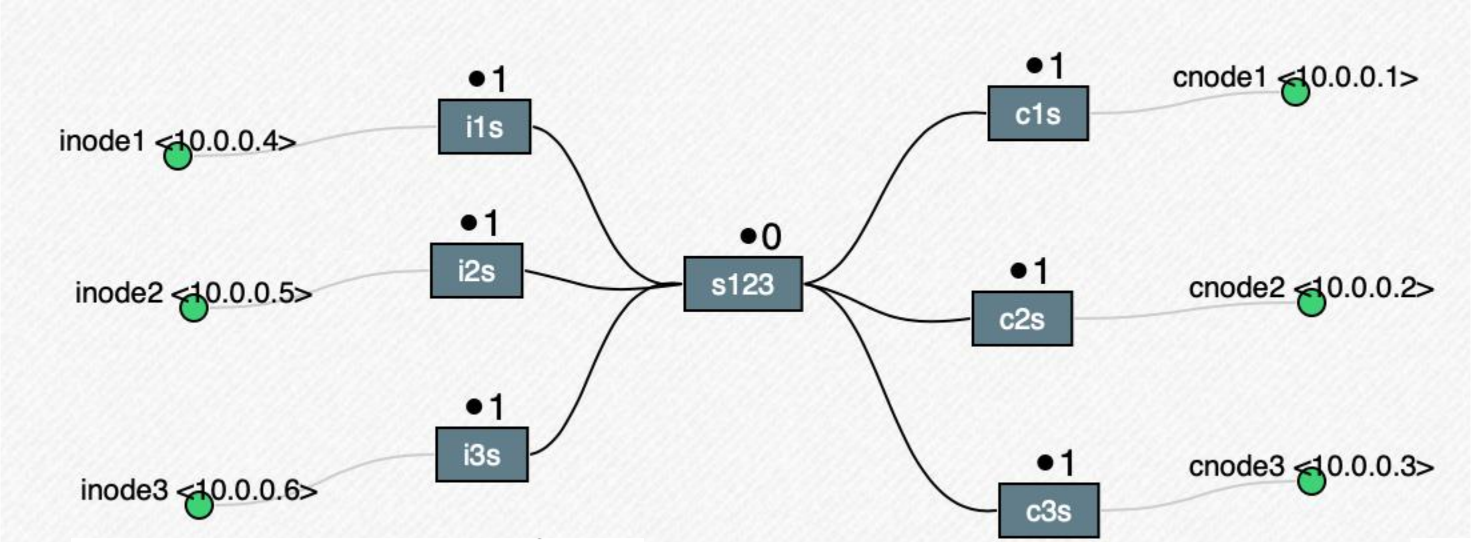
* on single core AMD64 architecture, 1 GB RAM (RAM usage: 173 MB)



Technical Approach (Implementation)

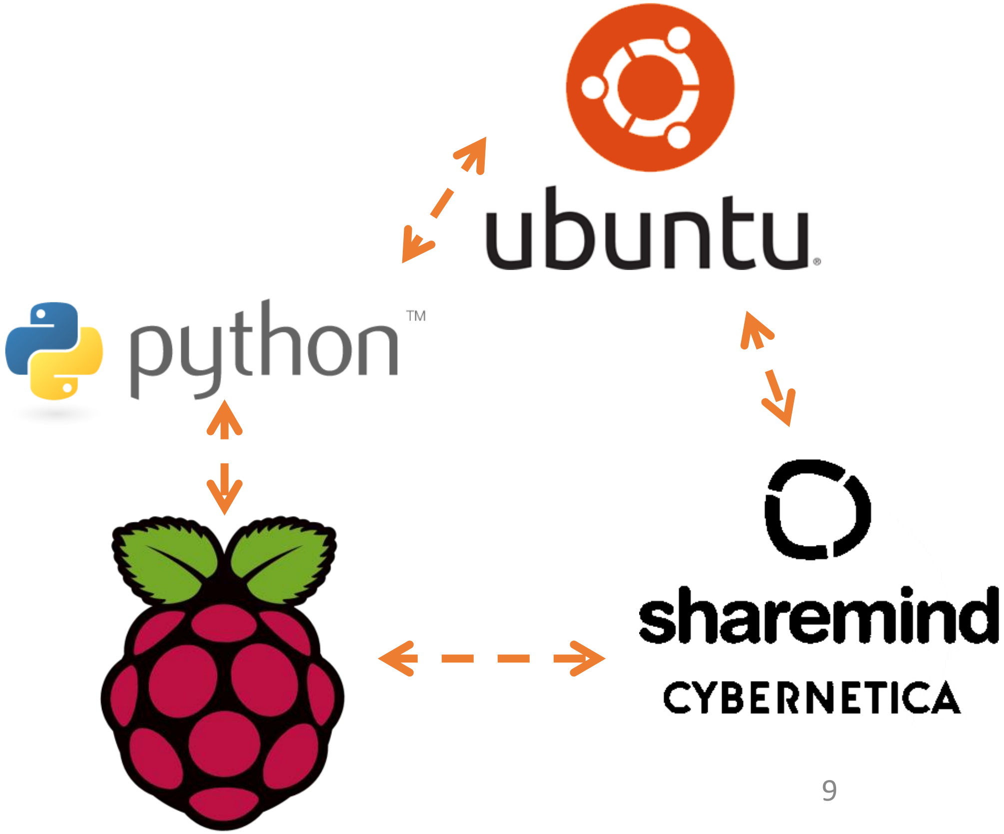
SDN Narmox Spear – Mininet

http://demo.spear.narmox.com/app/?apiurl=demo#!/mininet




```
('10.0.0.4', 12345): hello from comp
recv from comp
('10.0.0.4', 12345): hello from comp
recv from comp
('10.0.0.4', 12345): hello from
recv from comp
('10.0.0.1', 44635): hello from
('10.0.0.2', 41323): hello from
('10.0.0.3', 60280): hello from
('10.0.0.6', 12345): hello from
recv from comp
('10.0.0.6', 12345): hello from
recv from comp
('10.0.0.6', 12345): hello from
recv from comp
```

```
listening for incoming transmissions on 10.0.0.4
producing log files at log0.txt
listening for incoming transmissions on 10.0.0.5
producing log files at log1.txt
listening for incoming transmissions on 10.0.0.6
producing log files at log2.txt
inode: 0transmitting to 10.0.0.4
*** cnode1 : ('python inode.py -i 10.0.0.4 -m "hello from inode"',)
hello sent data
inode: 1transmitting to 10.0.0.4
*** cnode2 : ('python inode.py -i 10.0.0.4 -m "hello from inode"',)
hello sent data
inode: 2transmitting to 10.0.0.4
*** cnode3 : ('python inode.py -i 10.0.0.4 -m "hello from inode"',)
hello sent data
inode: 0transmitting to 10.0.0.5
*** cnode1 : ('python inode.py -i 10.0.0.5 -m "hello from inode"',)
hello sent data
inode: 1transmitting to 10.0.0.5
*** cnode2 : ('python inode.py -i 10.0.0.5 -m "hello from inode"',)
hello sent data
inode: 2transmitting to 10.0.0.5
*** cnode3 : ('python inode.py -i 10.0.0.5 -m "hello from inode"',)
```



Current Status and Next Steps

Current Status

- Aggregation and SMPC protocols designed and finalized
- Preliminary benchmarking of protocols and aggregation complete in virtual environment
- Simulator (Mininet/Sharemind emulator) and hardware (RPi with appropriate SDK) environment finalized and set up
- Oblivious custom operations implemented for aggregation and SMPC protocols

Next Steps

- Design more operations for aggregation and SMPC protocols.
- Implement the protocols in Mininet and RPi for real-time benchmarking metrics.
- Tune parameters to reduce computational overhead.

THANK YOU