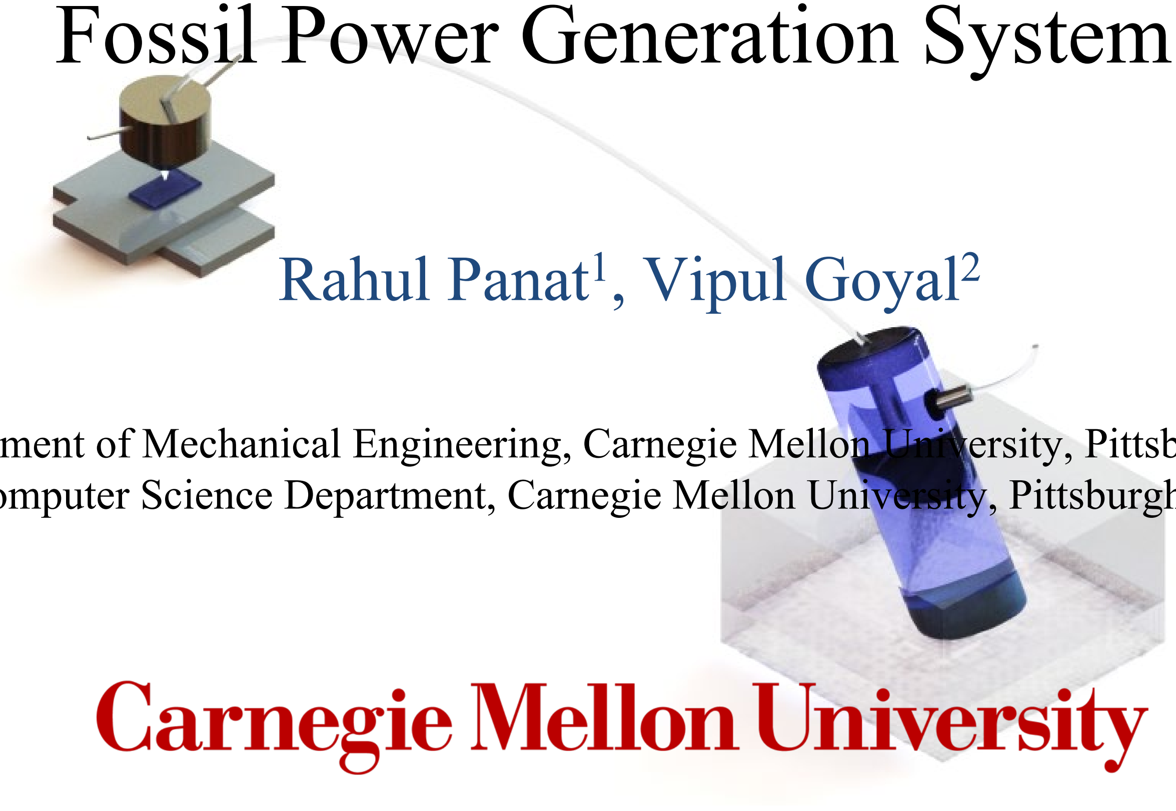
AOI 2: A Novel Access Control Blockchain Paradigm for Cybersecure Sensor Infrastructure in Fossil Power Generation Systems



Rahul Panat[1], Vipul Goyal[2]

[1]Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh PA
[2]Computer Science Department, Carnegie Mellon University, Pittsburgh PA

Outline

- Introduction and Background
  - Team
  - Project Goals and Objectives
  - Tasks and Timelines
- Building Cybersecure Sensor Networks
  - Strain Sensors
  - Temperature Sensors
- Private Access Controlled Blockchain
- Progress on Deliverables and Conclusions

The Team

Lab-scale Sensor Network

Rahul Panat
Project Lead PI

Blockchain Design and Coding

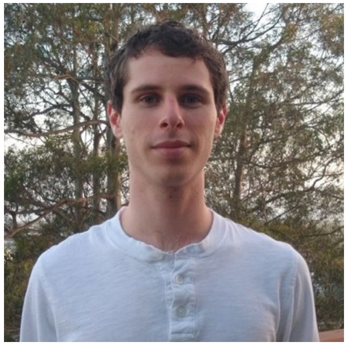
Vipul Goyal
Project Co-PI

Mrunal Vaze (MS)
Joined a Robotics
Company in Pittsburgh
(2020)

Sandra Ritchie
(PhD)

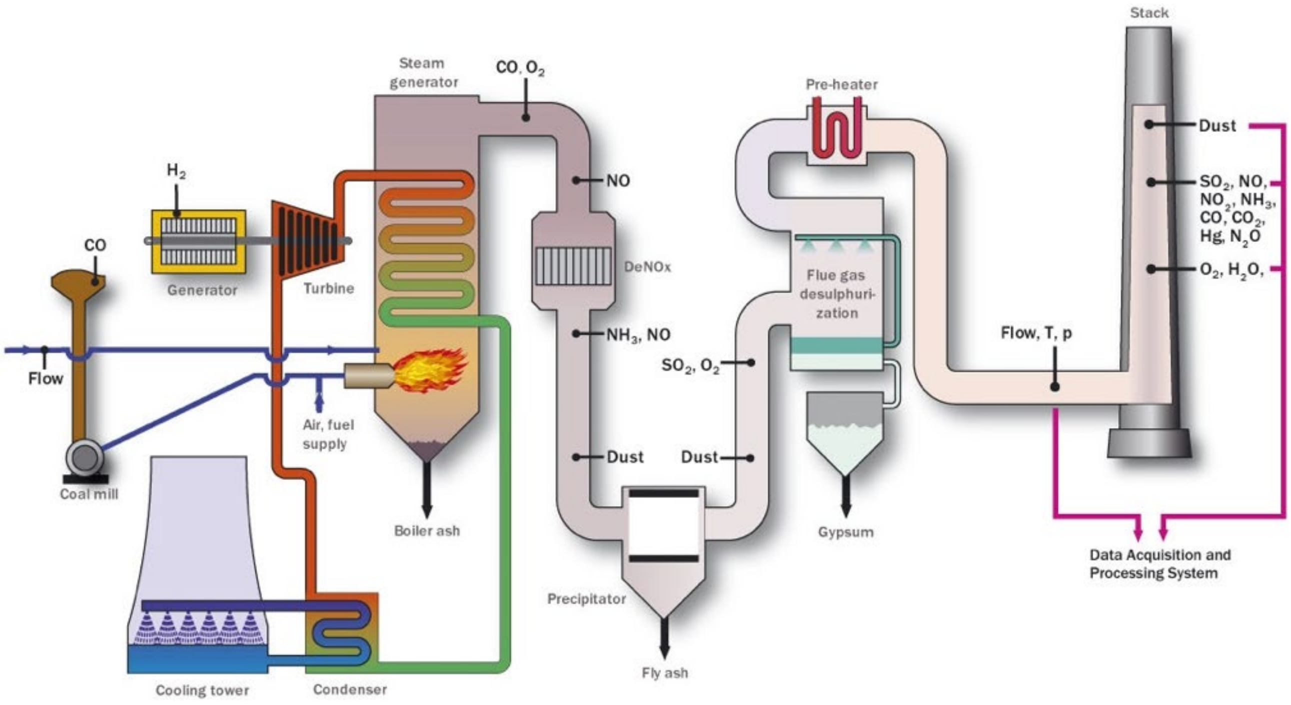
Elisaweta Masserova
(PhD)

Anirudh Baddepudi
(MS)



Justin Raizes
(PhD)

Sensing Applications

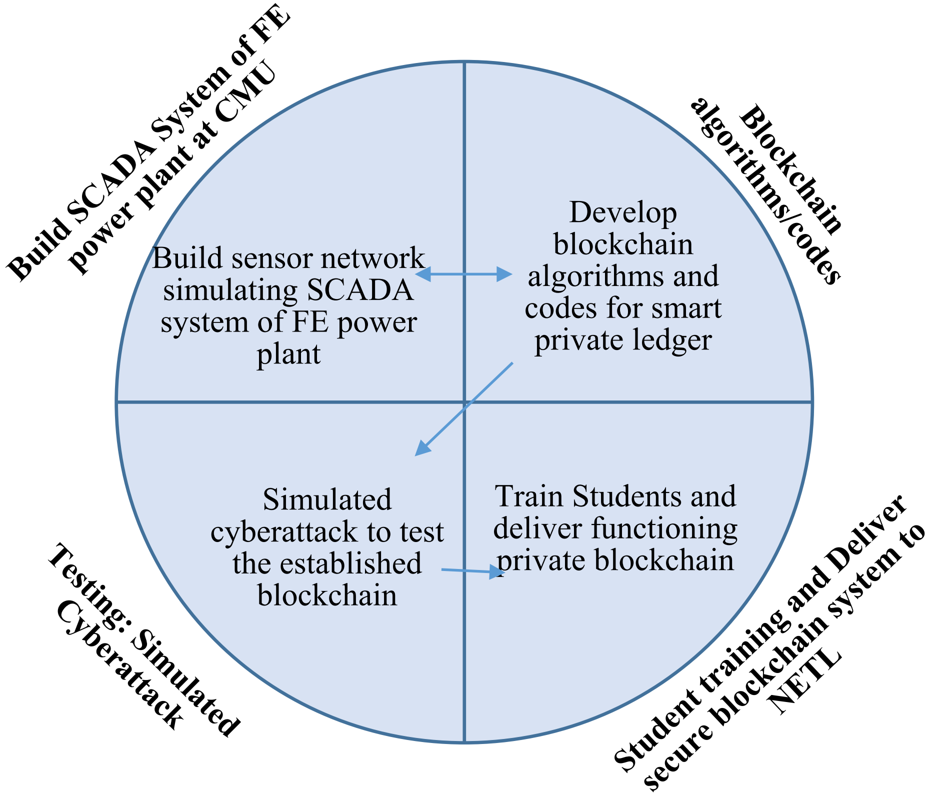


- Power generation and distribution infrastructure can experience both external or internal cyberattacks
- Novel methods are required to secure the data, while also controlling its access

Photo Source: S.R. Thermonix Technologies

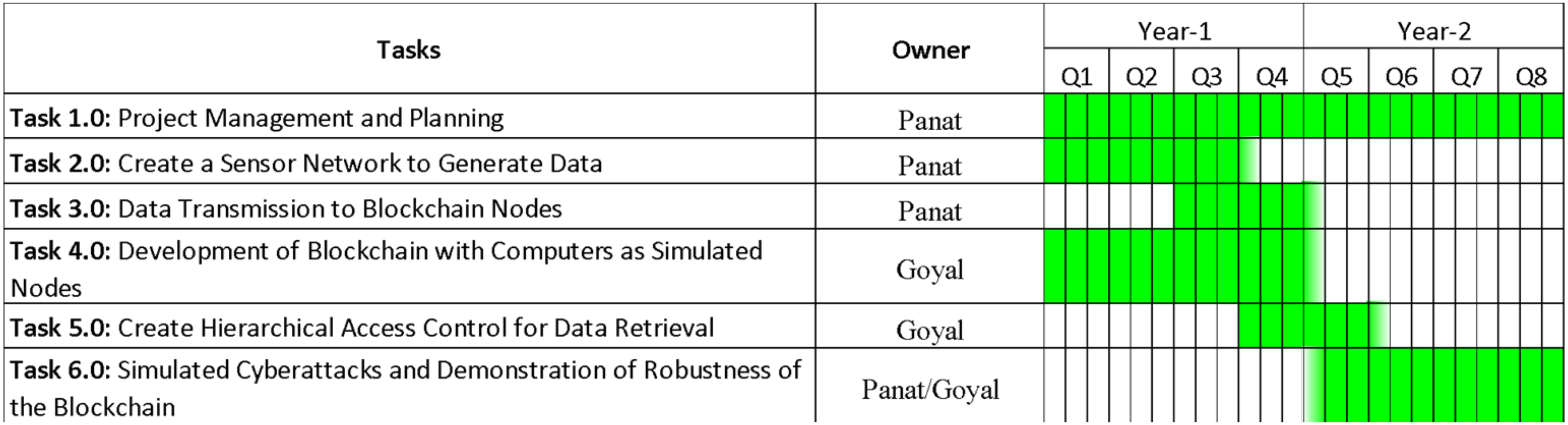
Objective of the Project

To design, characterize, and demonstrate a breakthrough secure blockchain protocol, namely smart private ledger with hierarchical access control for fossil power generation systems


Build SCADA System of FE power plant at CMU
Blockchain algorithms/codes
Build sensor network simulating SCADA system of FE power plant
Develop blockchain algorithms and codes for smart private ledger
Simulated cyberattack to test the established blockchain
Train Students and deliver functioning private blockchain
Testing: Simulated Cyberattack
Student training and Deliver secure blockchain system to NETL

Project Timelines and Deliverables

Tasks and Timelines



Tasks	Owner	Year-1												Year-2											
		Q1			Q2			Q3			Q4			Q5			Q6			Q7			Q8		
Task 1.0: Project Management and Planning	Panat	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Task 2.0: Create a Sensor Network to Generate Data	Panat	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩															
Task 3.0: Data Transmission to Blockchain Nodes	Panat							🟩	🟩	🟩	🟩	🟩	🟩												
Task 4.0: Development of Blockchain with Computers as Simulated Nodes	Goyal	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩												
Task 5.0: Create Hierarchical Access Control for Data Retrieval	Goyal										🟩	🟩	🟩	🟩	🟩	🟩									
Task 6.0: Simulated Cyberattacks and Demonstration of Robustness of the Blockchain	Panat/Goyal													🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

- Project period: 2 years
  - Data acquisition and transmission system
  - Creation of blockchain protocols
  - Simulate cyberattacks and demonstration lab-scale system

Task-1

- Project Management and Planning
  - The PIs will shall manage and direct the project in accordance with a Project Management Plan to meet all technical, schedule and budget objectives and requirements. The PIs will coordinate activities in order to effectively accomplish the work. The PIs will ensure that project plans, results, and decisions are appropriately documented and project reporting and briefing requirements are satisfied.

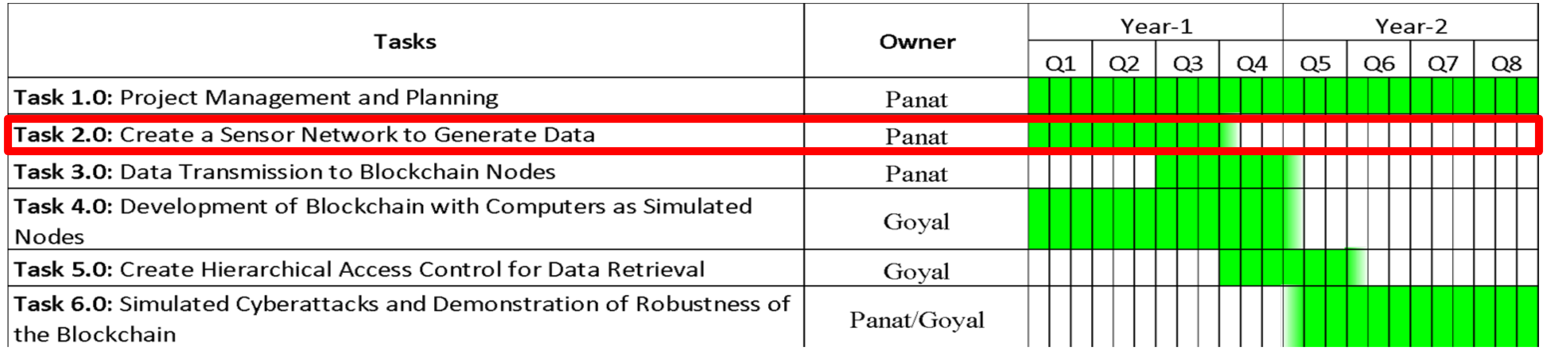
Tasks	Owner	Year-1												Year-2											
		Q1			Q2			Q3			Q4			Q5			Q6			Q7			Q8		
Task 1.0: Project Management and Planning	Panat	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Task 2.0: Create a Sensor Network to Generate Data	Panat	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩														
Task 3.0: Data Transmission to Blockchain Nodes	Panat							🟩	🟩	🟩	🟩	🟩	🟩	🟩											
Task 4.0: Development of Blockchain with Computers as Simulated Nodes	Goyal	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩											
Task 5.0: Create Hierarchical Access Control for Data Retrieval	Goyal										🟩	🟩	🟩	🟩	🟩	🟩	🟩								
Task 6.0: Simulated Cyberattacks and Demonstration of Robustness of the Blockchain	Panat/Goyal													🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

✓

ONGOING

Task-2

- Create a Sensor Network to Generate Data
  - This task will involve the development of sensor networks for the development of the proposed technology. The task will be performed by Panat group

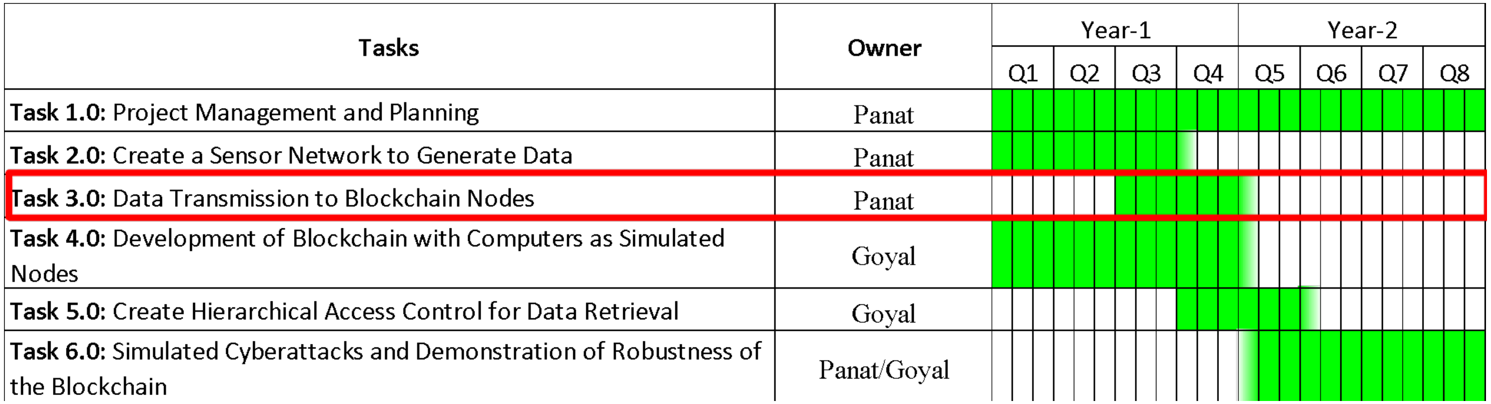


Tasks	Owner	Year-1				Year-2			
		Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Task 1.0: Project Management and Planning	Panat	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Task 2.0: Create a Sensor Network to Generate Data	Panat	🟩	🟩	🟩					
Task 3.0: Data Transmission to Blockchain Nodes	Panat			🟩	🟩				
Task 4.0: Development of Blockchain with Computers as Simulated Nodes	Goyal	🟩	🟩	🟩	🟩				
Task 5.0: Create Hierarchical Access Control for Data Retrieval	Goyal				🟩	🟩			
Task 6.0: Simulated Cyberattacks and Demonstration of Robustness of the Blockchain	Panat/Goyal					🟩	🟩	🟩	🟩

✓

Task-3

- Data Transmission to Blockchain Nodes
  - This task will involve the development of wireless transmission of the signal to the blockchain nodes. The task will be performed by Panat group

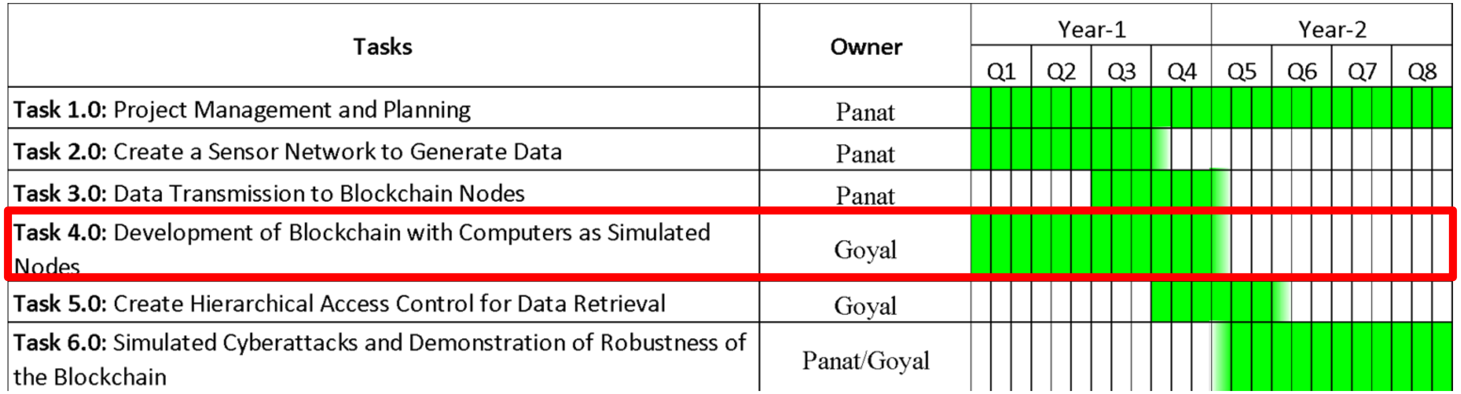


Tasks	Owner	Year-1												Year-2											
		Q1			Q2			Q3			Q4			Q5			Q6			Q7			Q8		
Task 1.0: Project Management and Planning	Panat	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Task 2.0: Create a Sensor Network to Generate Data	Panat	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩														
Task 3.0: Data Transmission to Blockchain Nodes	Panat							🟩	🟩	🟩	🟩	🟩	🟩	🟩											
Task 4.0: Development of Blockchain with Computers as Simulated Nodes	Goyal	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩											
Task 5.0: Create Hierarchical Access Control for Data Retrieval	Goyal										🟩	🟩	🟩	🟩	🟩	🟩	🟩								
Task 6.0: Simulated Cyberattacks and Demonstration of Robustness of the Blockchain	Panat/Goyal													🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

✓

Task-4

- Development of Blockchain with Computers as Simulated Nodes
  - This task will involve the development of the smart private ledger blockchain with hierarchical access control and secret sharing protocols and will be performed by the Goyal group.

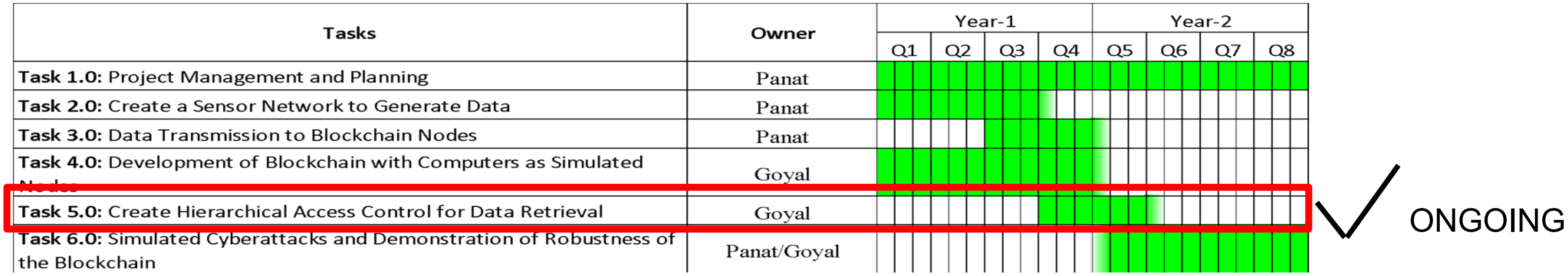


Tasks	Owner	Year-1												Year-2											
		Q1			Q2			Q3			Q4			Q5			Q6			Q7			Q8		
Task 1.0: Project Management and Planning	Panat	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Task 2.0: Create a Sensor Network to Generate Data	Panat	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩														
Task 3.0: Data Transmission to Blockchain Nodes	Panat							🟩	🟩	🟩	🟩	🟩	🟩	🟩											
Task 4.0: Development of Blockchain with Computers as Simulated Nodes	Goyal	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩											
Task 5.0: Create Hierarchical Access Control for Data Retrieval	Goyal										🟩	🟩	🟩	🟩	🟩	🟩	🟩								
Task 6.0: Simulated Cyberattacks and Demonstration of Robustness of the Blockchain	Panat/Goyal													🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

✓

Task-5

- Create Hierarchical Access Control for Data Retrieval
  - This task will develop algorithms to retrieve the data from the blockchain and will be performed by the Goyal group

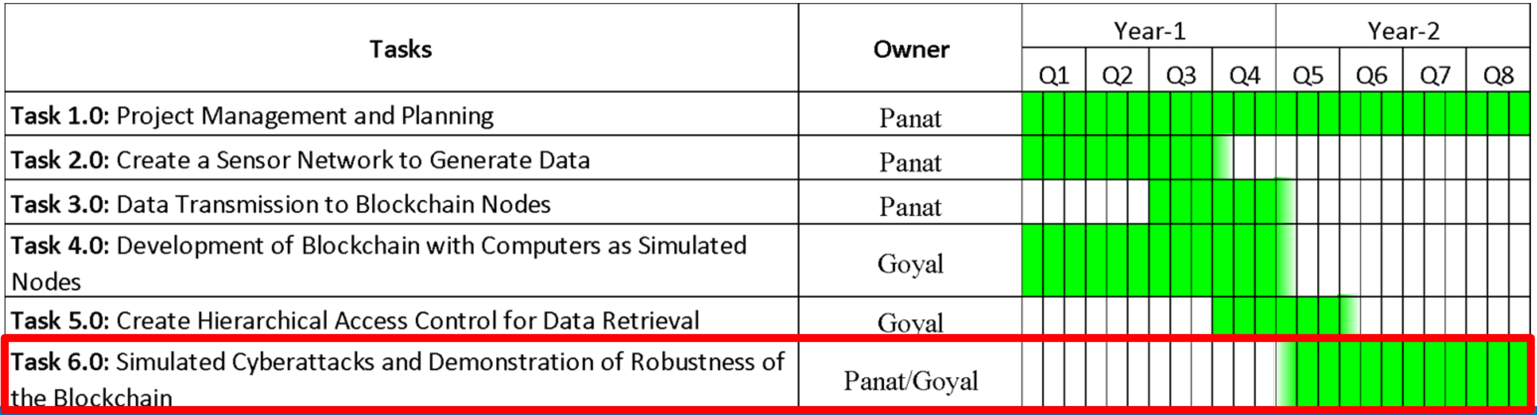


Tasks	Owner	Year-1												Year-2											
		Q1			Q2			Q3			Q4			Q5			Q6			Q7			Q8		
Task 1.0: Project Management and Planning	Panat	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Task 2.0: Create a Sensor Network to Generate Data	Panat	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩														
Task 3.0: Data Transmission to Blockchain Nodes	Panat							🟩	🟩	🟩	🟩	🟩	🟩	🟩											
Task 4.0: Development of Blockchain with Computers as Simulated Nodes	Goyal	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩											
Task 5.0: Create Hierarchical Access Control for Data Retrieval	Goyal										🟩	🟩	🟩	🟩	🟩	🟩	🟩								
Task 6.0: Simulated Cyberattacks and Demonstration of Robustness of the Blockchain	Panat/Goyal													🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

✓ ONGOING

Task-6

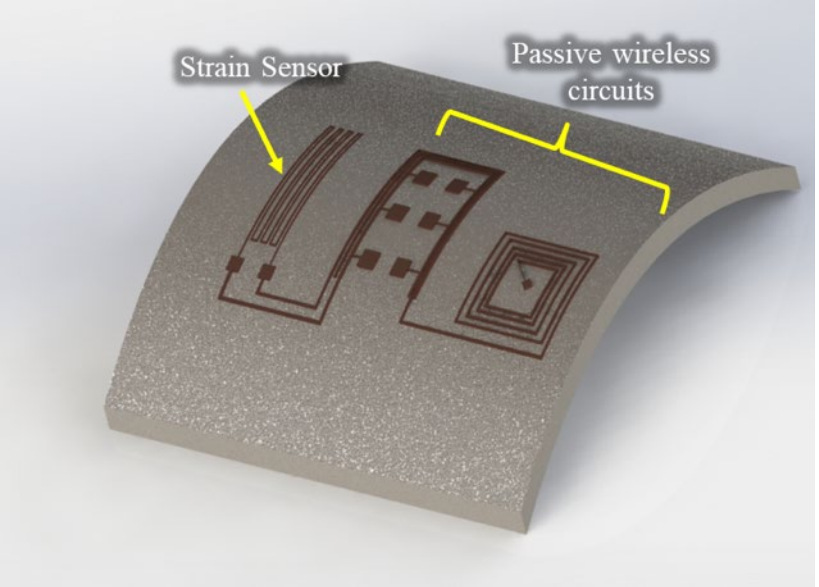
- Simulated Cyberattacks and Demonstration of Robustness of the Blockchain
  - PIs will simulate cyberattacks to harden the blockchain system for real world secure deployment
  - Common strategies such as those used during the Ukranian power grid attack will be studied and the blockchain system will be subjected to similar attacks.
  - Any changes if needed will be made and the entire process will be repeated. We expect our system to provide very high level of security against such attacks by eliminating a single point of failure.

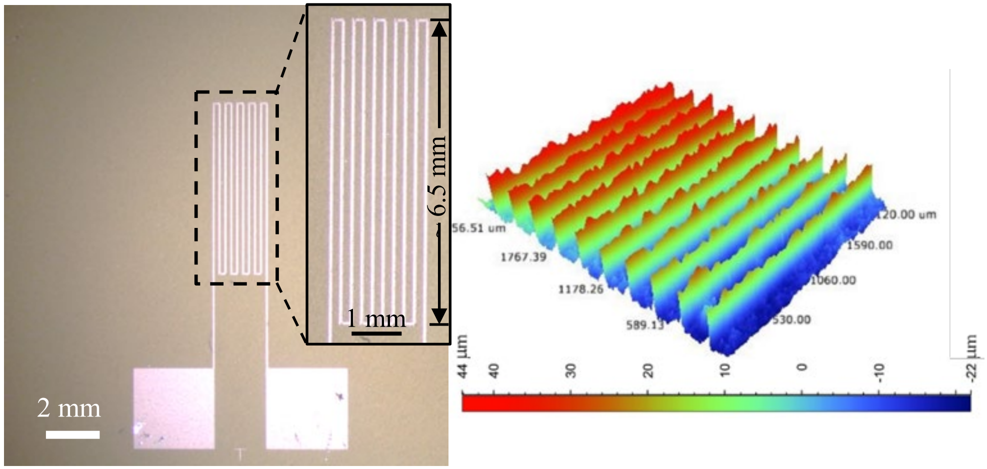


Tasks	Owner	Year-1												Year-2											
		Q1			Q2			Q3			Q4			Q5			Q6			Q7			Q8		
Task 1.0: Project Management and Planning	Panat	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩
Task 2.0: Create a Sensor Network to Generate Data	Panat	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩														
Task 3.0: Data Transmission to Blockchain Nodes	Panat							🟩	🟩	🟩	🟩	🟩	🟩	🟩											
Task 4.0: Development of Blockchain with Computers as Simulated Nodes	Goyal	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩											
Task 5.0: Create Hierarchical Access Control for Data Retrieval	Goyal										🟩	🟩	🟩	🟩	🟩	🟩	🟩								
Task 6.0: Simulated Cyberattacks and Demonstration of Robustness of the Blockchain	Panat/Goyal													🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩

Building Sensor Network

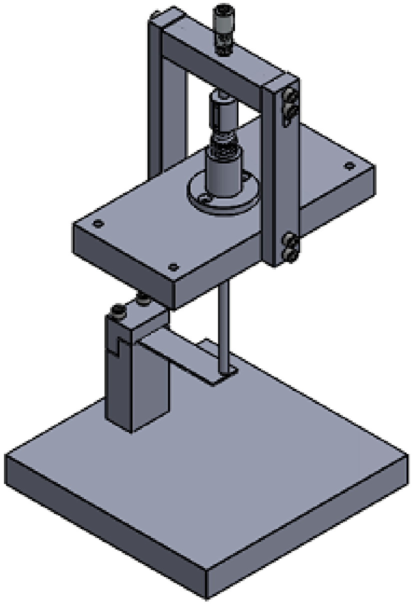
High Temperature Sensor Fabrication

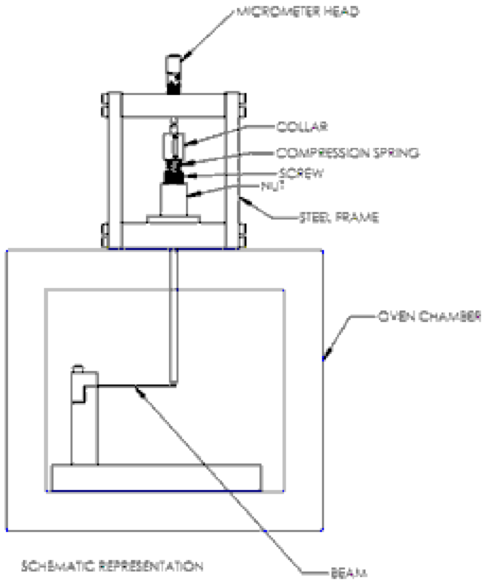


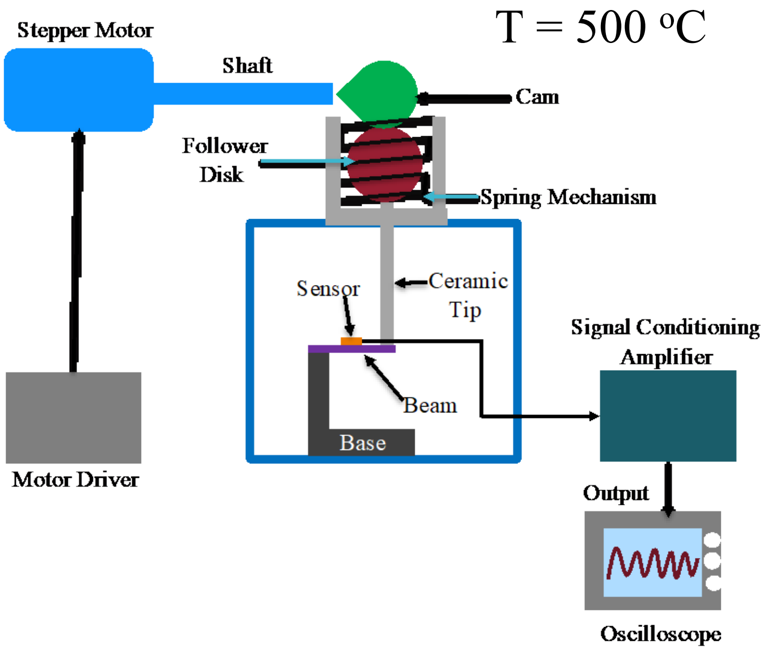


CMU has developed sensor fabrication methods and testing systems for fossil power plants that can work at temperatures up to 500 C

High Temperature Sensor Testing

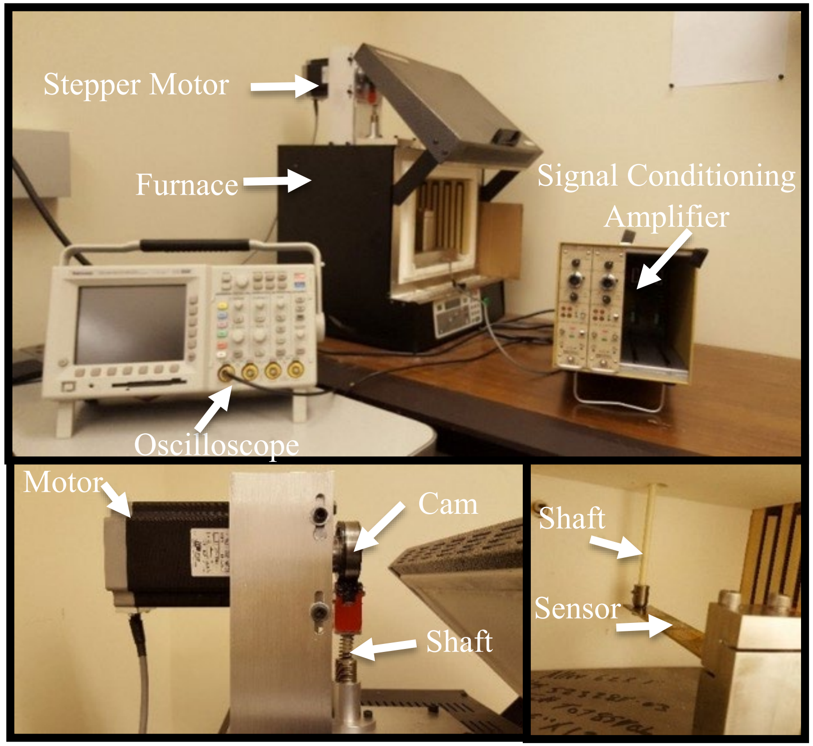




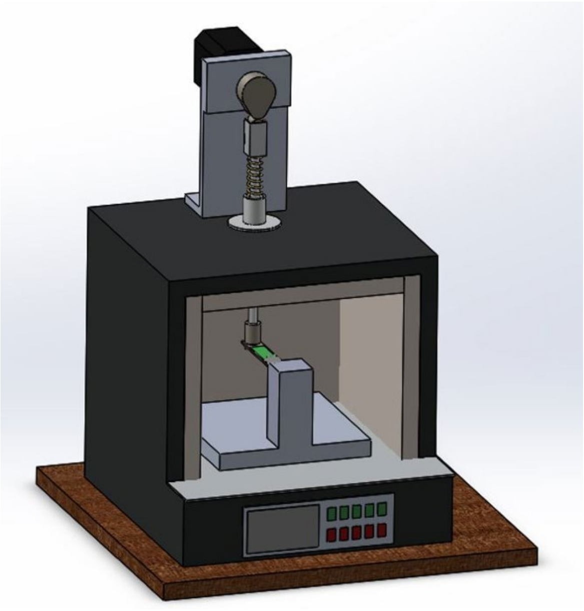


Schematic of the Strain Sensing Apparatus

High Temperature Data Acquisition System



High Temperature Dynamic
Strain Sensor Test Set up



- Able to provide 1000 micro strain on the beam
- Deflection frequency: up to 10 Hz

Strain Measurement Apparatus

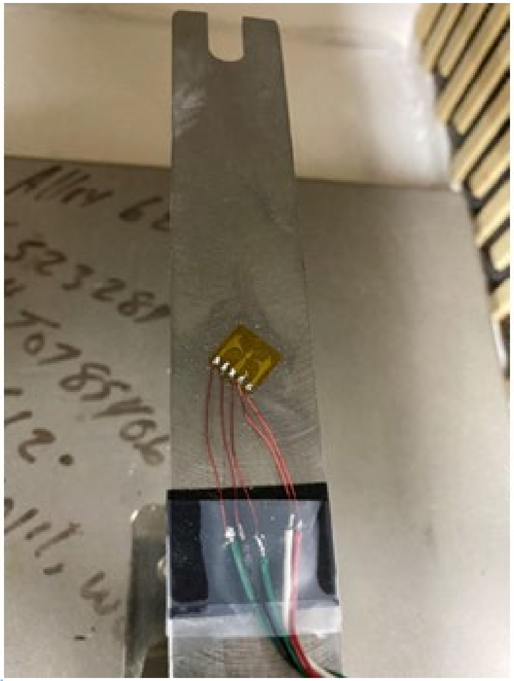
Strain Measurement

Successfully demonstrated strain measurement using Mantracourt T24 telemetry system

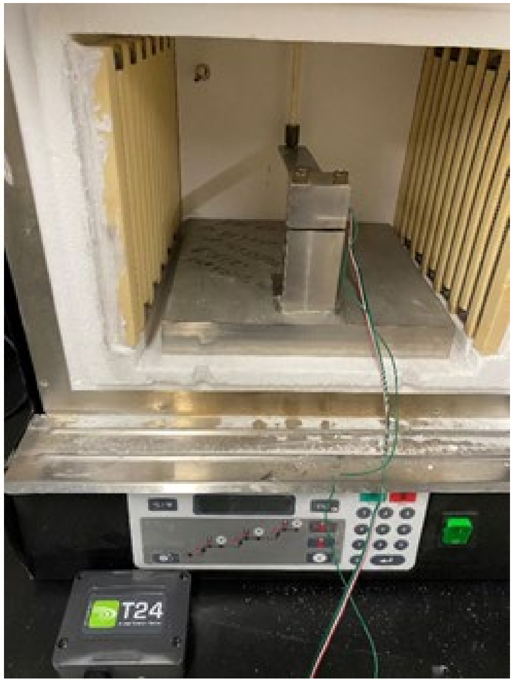
- Installed a commercial strain sensor (VY4 Shear/Torsion full bridge strain gauge) acquired from HBM, USA
- Integrated the strain sensor with transmitter and base station
- Data acquisition at 3 readings/sec – compatible with power plant sensing systems

Strain Measurement

Stainless steel beam

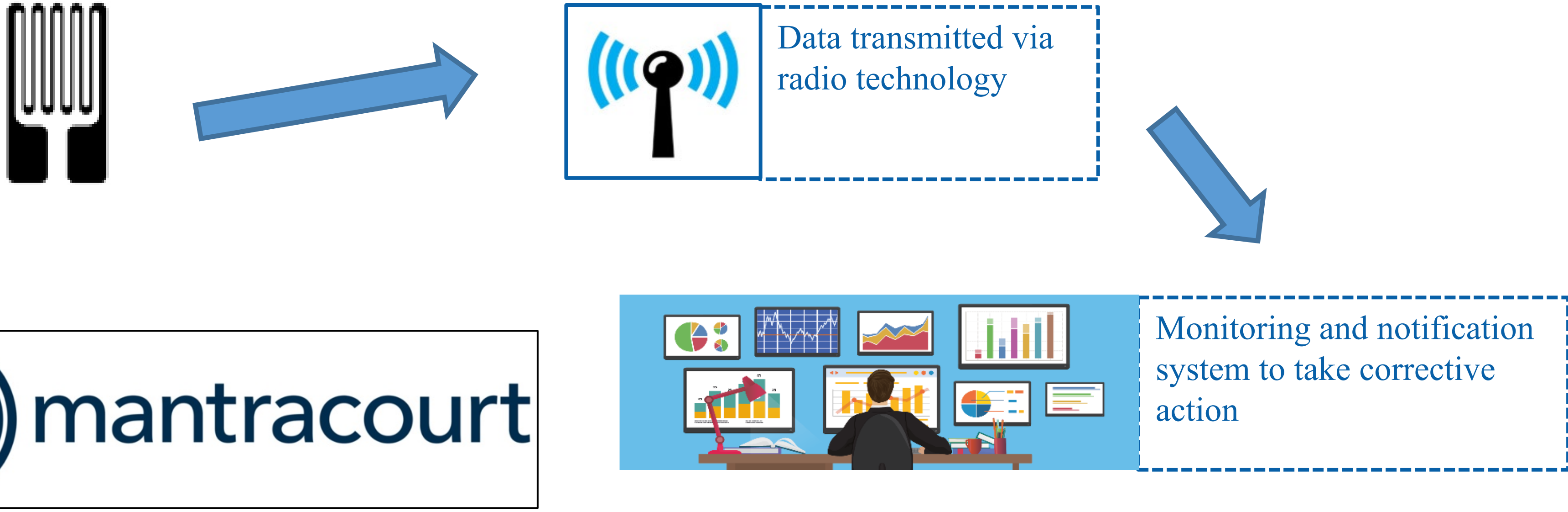


Strain sensor showing good adhesion to beam surface



Strain sensor integrated with transmitter module

Data Transmission: Mantracourt System

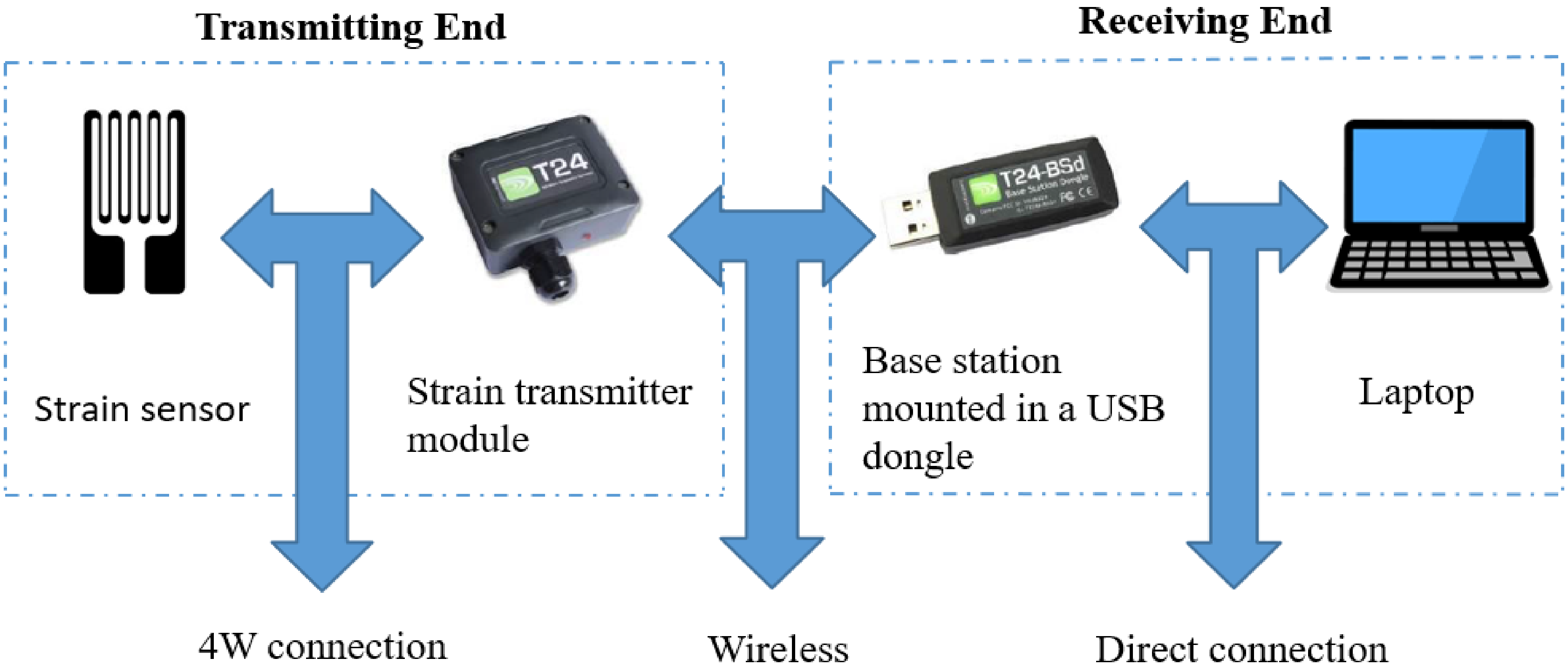


Data transmitted via radio technology

Monitoring and notification system to take corrective action

- Chose mantracourt system for secure data transmission
- Commercially available system with low cost
- Aim was to create software compatible with commercial technologies for adaptability and lowering of cost

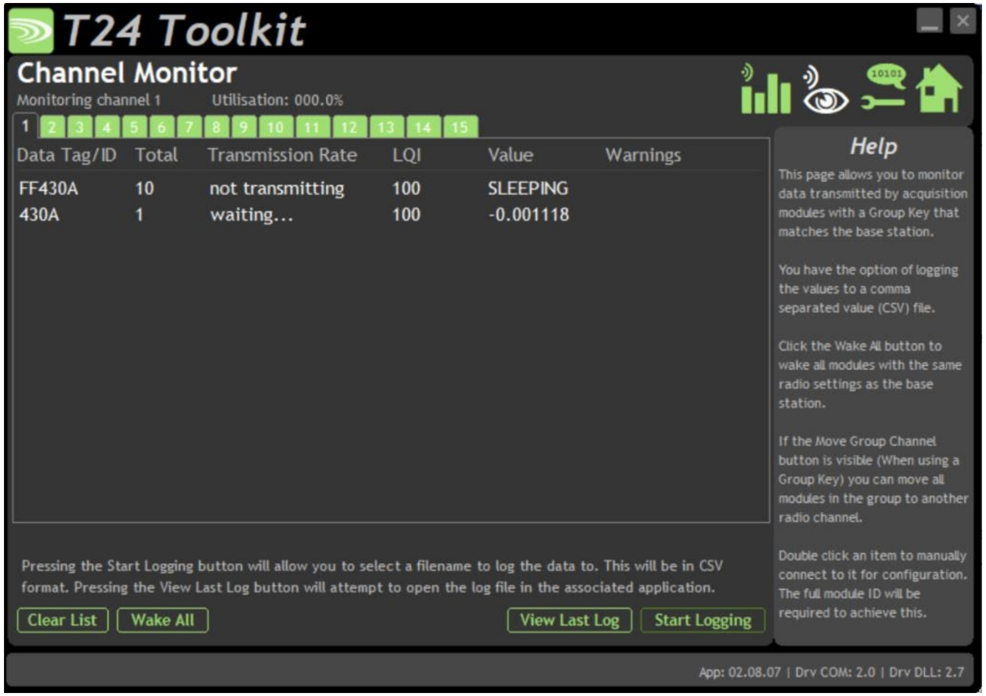
Data Transmission: Mantracourt System



- All types of sensors can be attached to the system reading voltage or current
- 600 m range in an open field site w/ license free 2.4 GHz direct sequence spread spectrum (DSSS) radio technology
- Data Encryption for complete security (128-bit AES)
- Proprietary protocol based on 802.15.4 chip allowing T24 range to co-exist with Bluetooth, Zigbee & Wi-Fi devices w/o conflicts

Data Transmission

- Blockchain coding required the data to be in readable txt format
- One transmitter can be connected to up to 15 sensors – data transmitted to a USB base station connected to a computer in .csv file
- Frequency control to save power with this platform



Monitoring channel 1 Utilisation: 000.0%

Data Tag/ID	Total	Transmission Rate	LQI	Value	Warnings
FF430A	10	not transmitting	100	SLEEPING	
430A	1	waiting...	100	-0.001118	

Pressing the Start Logging button will allow you to select a filename to log the data to. This will be in CSV format. Pressing the View Last Log button will attempt to open the log file in the associated application.

This page allows you to monitor data transmitted by acquisition modules with a Group Key that matches the base station.

You have the option of logging the values to a comma separated value (CSV) file.

Click the Wake All button to wake all modules with the same radio settings as the base station.

If the Move Group Channel button is visible (When using a Group Key) you can move all modules in the group to another radio channel.

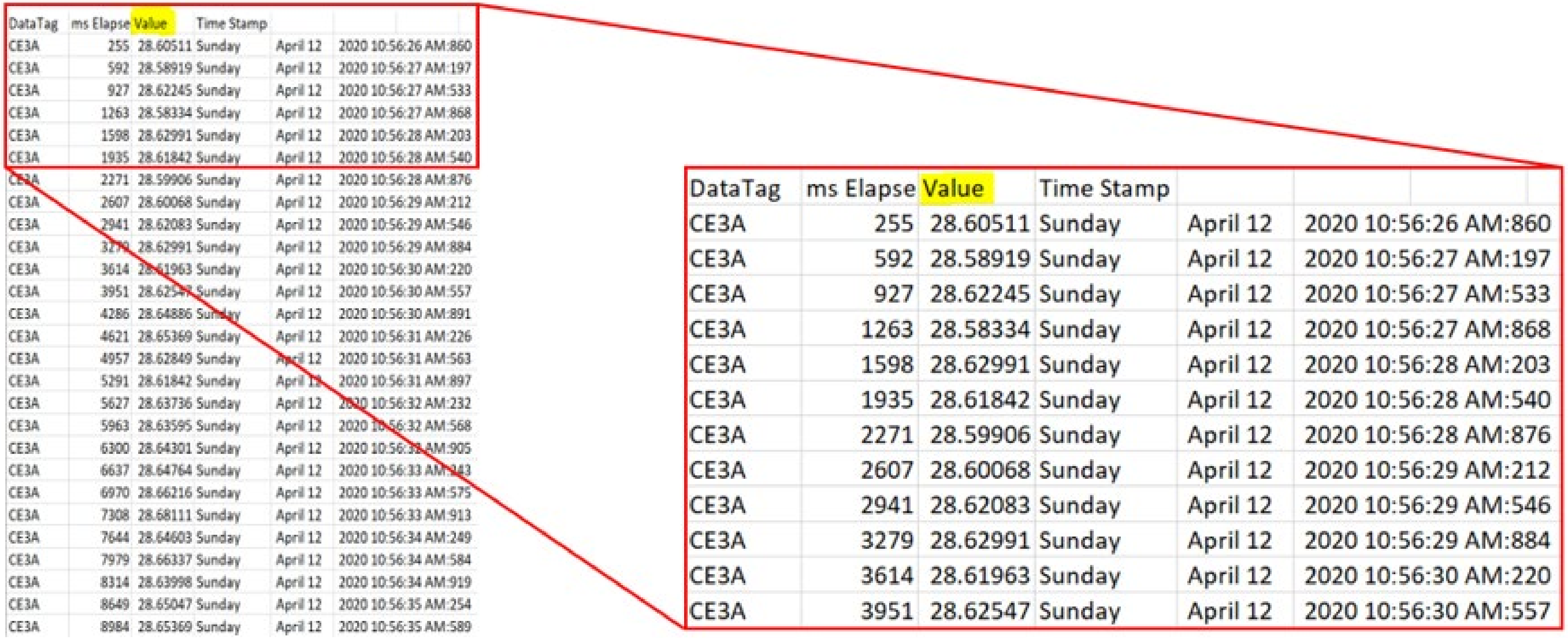
Double click an item to manually connect to it for configuration. The full module ID will be required to achieve this.

App: 02.08.07 | Drv COM: 2.0 | Drv DLL: 2.7

Temperature Measurement

- We chose commercial RTD temperature sensor for the project
- Temperature sensor integrated with Mantracourt T24 acquisition and wireless transmission system

Example: Temperature Measurement



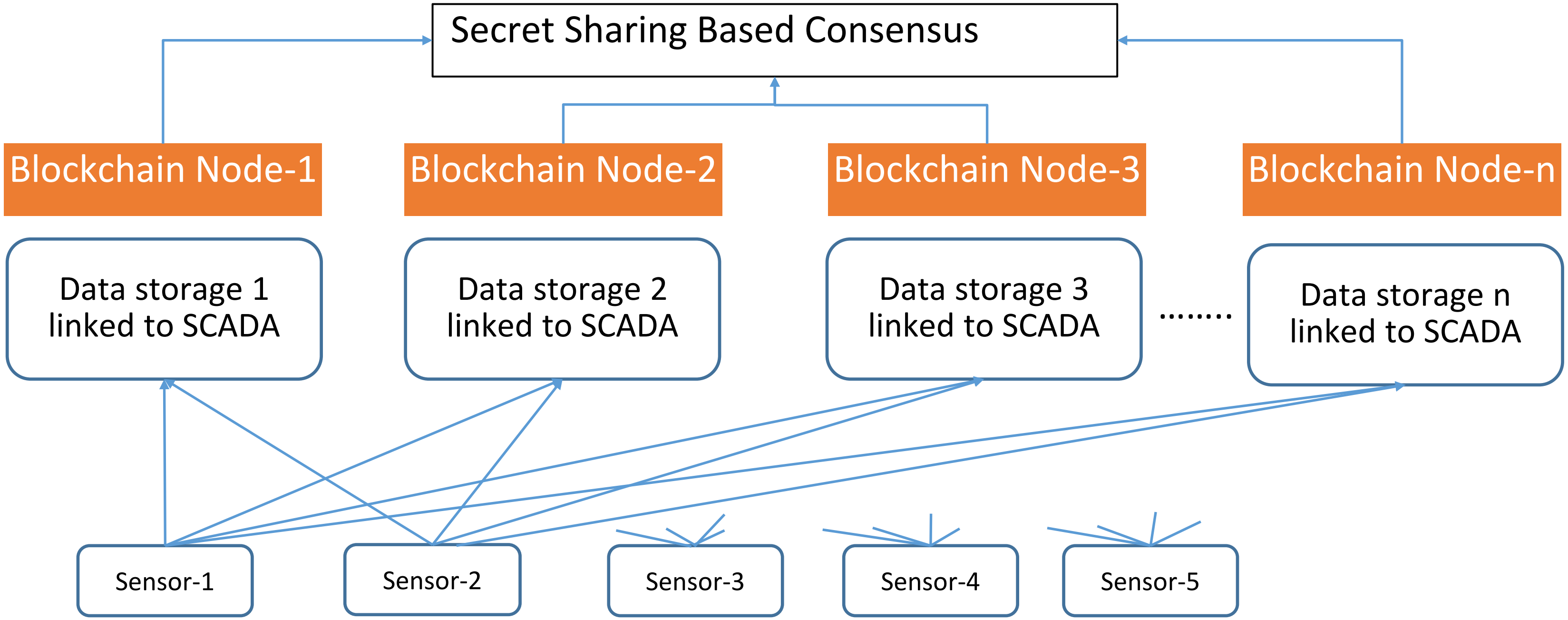
DataTag	ms Elapse	Value	Time Stamp		
CE3A	255	28.60511	Sunday	April 12	2020 10:56:26 AM:860
CE3A	592	28.58919	Sunday	April 12	2020 10:56:27 AM:197
CE3A	927	28.62245	Sunday	April 12	2020 10:56:27 AM:533
CE3A	1263	28.58334	Sunday	April 12	2020 10:56:27 AM:868
CE3A	1598	28.62991	Sunday	April 12	2020 10:56:28 AM:203
CE3A	1935	28.61842	Sunday	April 12	2020 10:56:28 AM:540
CE3A	2271	28.59906	Sunday	April 12	2020 10:56:28 AM:876
CE3A	2607	28.60068	Sunday	April 12	2020 10:56:29 AM:212
CE3A	2941	28.62083	Sunday	April 12	2020 10:56:29 AM:546
CE3A	3279	28.62991	Sunday	April 12	2020 10:56:29 AM:884
CE3A	3614	28.61963	Sunday	April 12	2020 10:56:30 AM:220
CE3A	3951	28.62547	Sunday	April 12	2020 10:56:30 AM:557
CE3A	4286	28.64886	Sunday	April 12	2020 10:56:30 AM:891
CE3A	4621	28.65369	Sunday	April 12	2020 10:56:31 AM:226
CE3A	4957	28.62849	Sunday	April 12	2020 10:56:31 AM:563
CE3A	5291	28.61842	Sunday	April 12	2020 10:56:31 AM:897
CE3A	5627	28.63736	Sunday	April 12	2020 10:56:32 AM:232
CE3A	5963	28.63595	Sunday	April 12	2020 10:56:32 AM:568
CE3A	6300	28.64301	Sunday	April 12	2020 10:56:32 AM:905
CE3A	6637	28.64764	Sunday	April 12	2020 10:56:33 AM:243
CE3A	6970	28.66216	Sunday	April 12	2020 10:56:33 AM:575
CE3A	7308	28.68111	Sunday	April 12	2020 10:56:33 AM:913
CE3A	7644	28.64603	Sunday	April 12	2020 10:56:34 AM:249
CE3A	7979	28.66337	Sunday	April 12	2020 10:56:34 AM:584
CE3A	8314	28.63998	Sunday	April 12	2020 10:56:34 AM:919
CE3A	8649	28.65047	Sunday	April 12	2020 10:56:35 AM:254
CE3A	8984	28.65369	Sunday	April 12	2020 10:56:35 AM:589

DataTag	ms Elapse	Value	Time Stamp		
CE3A	255	28.60511	Sunday	April 12	2020 10:56:26 AM:860
CE3A	592	28.58919	Sunday	April 12	2020 10:56:27 AM:197
CE3A	927	28.62245	Sunday	April 12	2020 10:56:27 AM:533
CE3A	1263	28.58334	Sunday	April 12	2020 10:56:27 AM:868
CE3A	1598	28.62991	Sunday	April 12	2020 10:56:28 AM:203
CE3A	1935	28.61842	Sunday	April 12	2020 10:56:28 AM:540
CE3A	2271	28.59906	Sunday	April 12	2020 10:56:28 AM:876
CE3A	2607	28.60068	Sunday	April 12	2020 10:56:29 AM:212
CE3A	2941	28.62083	Sunday	April 12	2020 10:56:29 AM:546
CE3A	3279	28.62991	Sunday	April 12	2020 10:56:29 AM:884
CE3A	3614	28.61963	Sunday	April 12	2020 10:56:30 AM:220
CE3A	3951	28.62547	Sunday	April 12	2020 10:56:30 AM:557

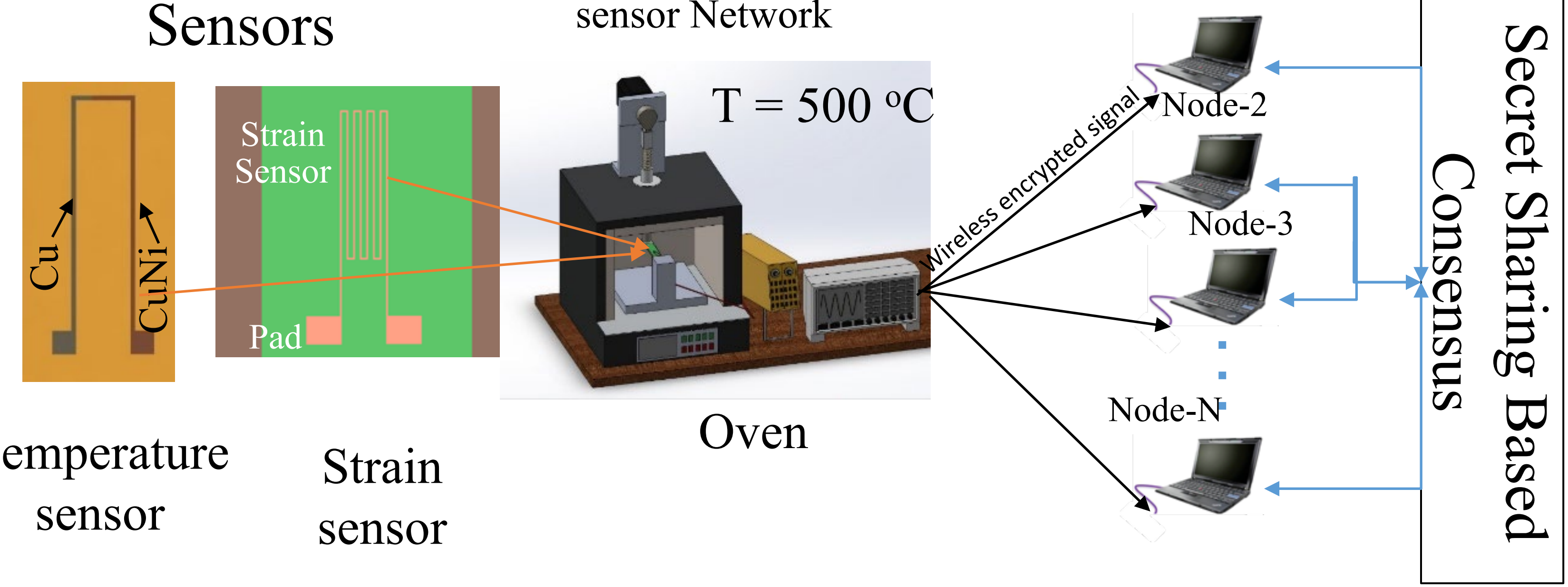
- Snapshot of temperature data collected in a .csv file
- This data directly feeds into the smart private ledger blockchain as discussed next

Smart Private Ledger: Blockchains with Private Computation

The Overall Vision: Create Smart Private Ledger


Secret Sharing Based Consensus
Blockchain Node-1
Blockchain Node-2
Blockchain Node-3
Blockchain Node-n
Data storage 1 linked to SCADA
Data storage 2 linked to SCADA
Data storage 3 linked to SCADA
....
Data storage n linked to SCADA
Sensor-1
Sensor-2
Sensor-3
Sensor-4
Sensor-5

Integration in Data Acquisition System



Smart private ledger Blockchain

Sensors

Oven for High Temperature sensor Network

Temperature sensor

Strain sensor

Oven

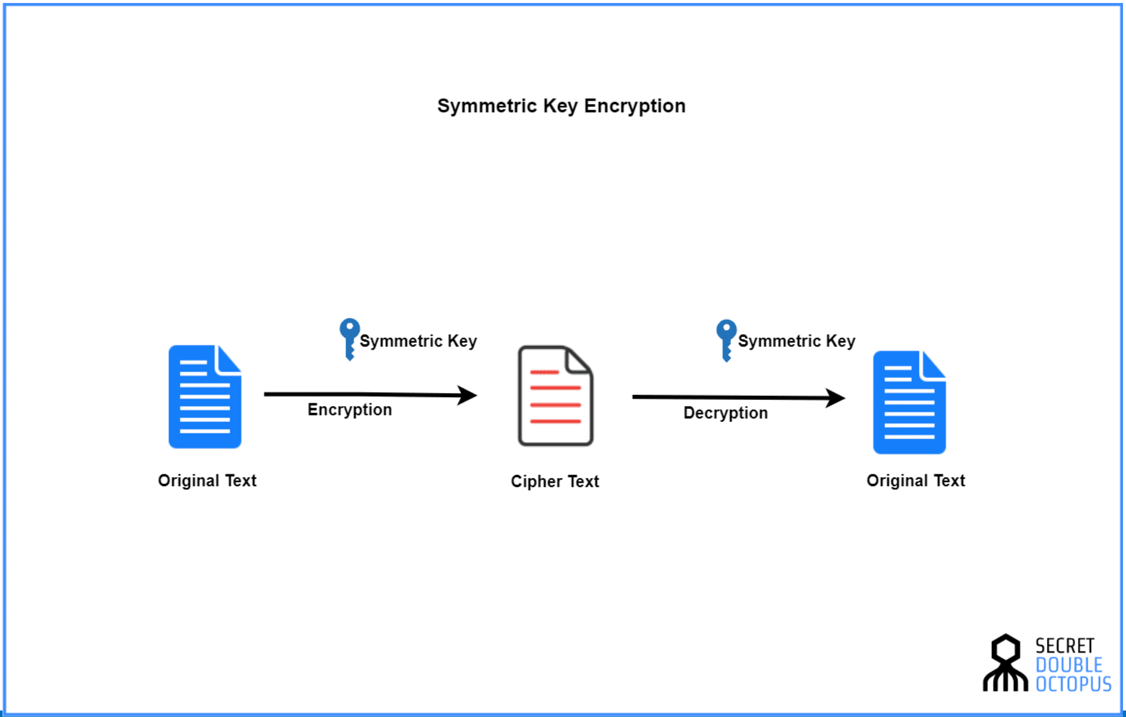
Need for Private Data

- As of today:
  - *All data on public ledger = public*
  - Private, access controlled data?
- Build an intelligent access controlled ledger
  - Different data visible to different parties
  - Even do computation on private data
  - 3rd gen Blockchain tech

Development of Smart Private Ledger

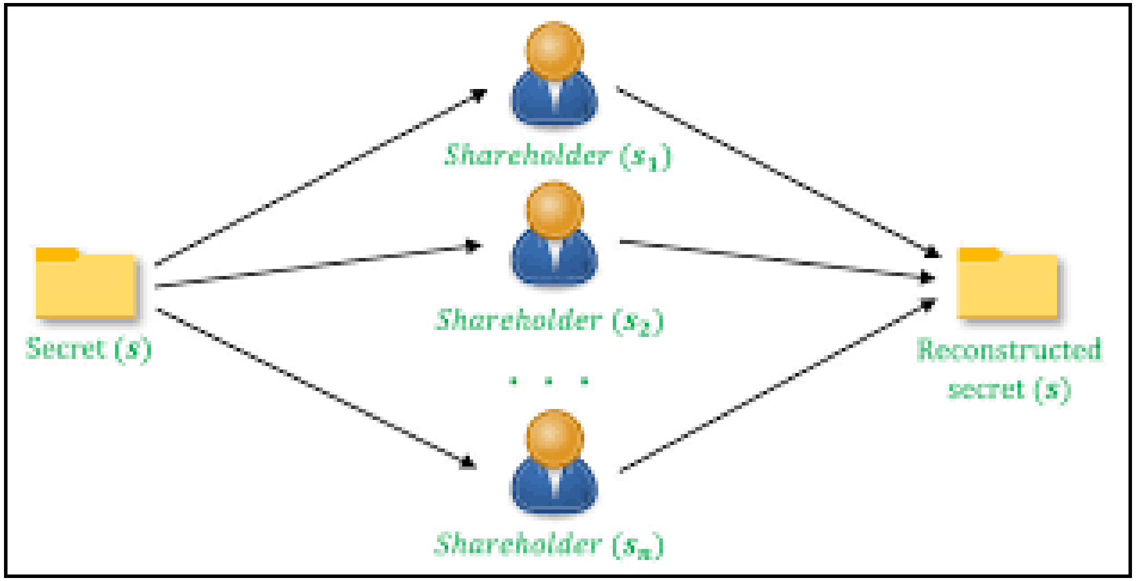
Our system flow is as follows:

- Generating secret key (for efficiency)
- Loading and encrypting csv file containing the data from sensor network (using AES)



Development of Smart Private Ledger

- Generating secret key shares
- Encrypting shares (using RSA) under miner public keys
- Later: decrypting secret key shares
- Reconstructing secret key
- Decrypting ciphertext to obtain original file containing data
- Smart contract to store/retrieve data from blockchain



Current completed components

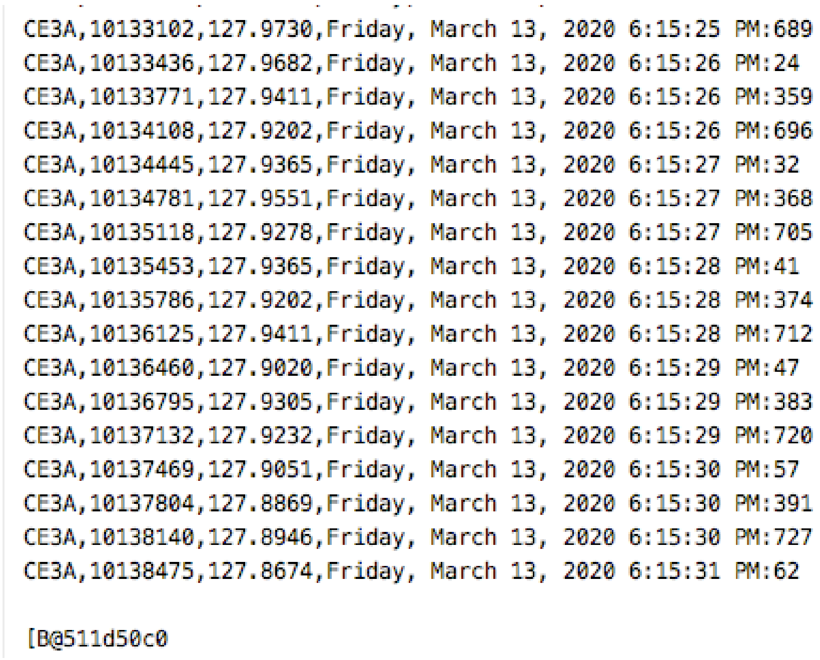
- The secret sharing and file encryption code is implemented to be run locally on a given miner's machine. The current implementation includes the following:
  - Generating secret key
  - Loading/encrypting csv file (using AES)
  - Generating secret key shares
  - Encrypting secret key shares (using RSA)
  - Decrypting secret key shares
  - Reconstructing secret key
  - Decrypting ciphertext to obtain original CSV file
  - Smart contract to store/retrieve data from blockchain

System Design

- Secret sharing and file encryption is implemented to be run locally on a given miner’s machine.
- Once this data is generated, it is stored in the smart contract which is deployed on the blockchain (Ethereum).
- Any miner is then able to access the data from the smart contract, decrypt their respective shares

Loading in the CSV File

- We first load in the CSV file and convert it to byte[] form. Pictures of this are shown below:

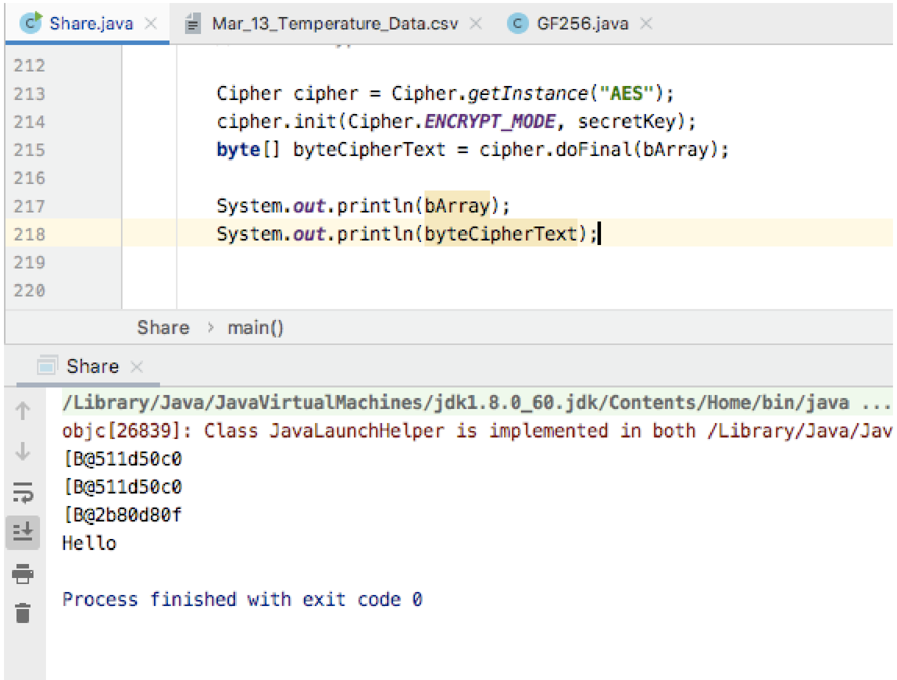


```
CE3A,10133102,127.9730,Friday, March 13, 2020 6:15:25 PM:689
CE3A,10133436,127.9682,Friday, March 13, 2020 6:15:26 PM:24
CE3A,10133771,127.9411,Friday, March 13, 2020 6:15:26 PM:359
CE3A,10134108,127.9202,Friday, March 13, 2020 6:15:26 PM:696
CE3A,10134445,127.9365,Friday, March 13, 2020 6:15:27 PM:32
CE3A,10134781,127.9551,Friday, March 13, 2020 6:15:27 PM:368
CE3A,10135118,127.9278,Friday, March 13, 2020 6:15:27 PM:705
CE3A,10135453,127.9365,Friday, March 13, 2020 6:15:28 PM:41
CE3A,10135786,127.9202,Friday, March 13, 2020 6:15:28 PM:374
CE3A,10136125,127.9411,Friday, March 13, 2020 6:15:28 PM:712
CE3A,10136460,127.9020,Friday, March 13, 2020 6:15:29 PM:47
CE3A,10136795,127.9305,Friday, March 13, 2020 6:15:29 PM:383
CE3A,10137132,127.9232,Friday, March 13, 2020 6:15:29 PM:720
CE3A,10137469,127.9051,Friday, March 13, 2020 6:15:30 PM:57
CE3A,10137804,127.8869,Friday, March 13, 2020 6:15:30 PM:391
CE3A,10138140,127.8946,Friday, March 13, 2020 6:15:30 PM:727
CE3A,10138475,127.8674,Friday, March 13, 2020 6:15:31 PM:62

[B@511d50c0
```

- Above is the CSV file, and below is the converted byte[] form. We require the file to be in this format for encryption/decryption, and will be able to convert back as shown later.

Generating the secret key

- The next step is to generate the secret key and encrypt the CSV file (converted to byte[] form) using the secret key. A picture of this code execution is shown below:
- We use the AES symmetric encryption scheme for file encryption/decryption.



```
        Cipher cipher = Cipher.getInstance("AES");
        cipher.init(Cipher.ENCRYPT_MODE, secretKey);
        byte[] byteCipherText = cipher.doFinal(bArray);

        System.out.println(bArray);
        System.out.println(byteCipherText);
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/java ...
objc[26839]: Class JavaLaunchHelper is implemented in both /Library/Java/Jav
[B@511d50c0
[B@511d50c0
[B@2b80d80f
Hello

Process finished with exit code 0
```

The first two byte[] values are the original file (bArray), and the third is the encrypted version (byteCipherText).

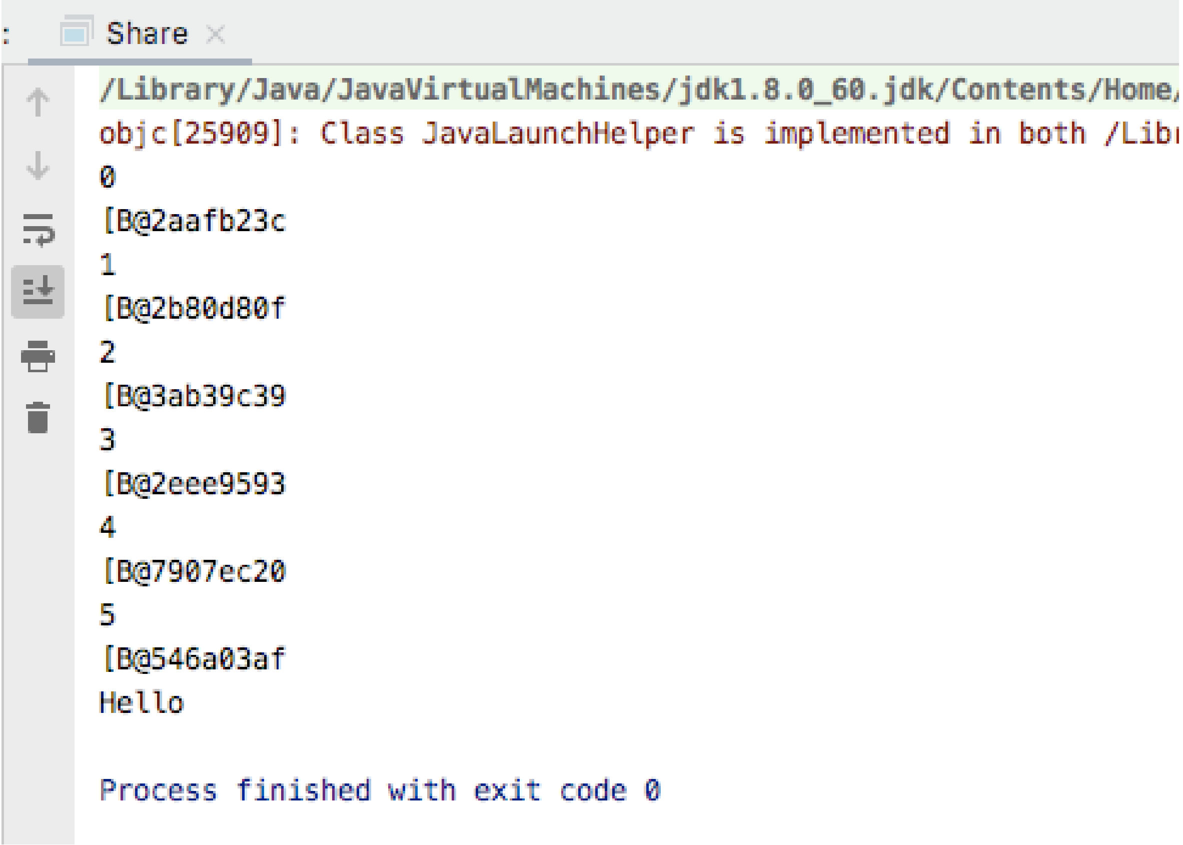
Secret Sharing

- We implement a function that generates the shares and reconstructs the secret key given the shares. The shares are output as a HashMap.

- The Dealer (person who owns the secret) does the following in order:
  1) Encrypts the data file using a generated secret key
  2) Generates the shares of the secret key using the Shamir secret sharing scheme
  3) Signs the shares so that we are able to identify dishonest miners
  4) Encrypts the shares using the corresponding miner public keys
  5) Posts the encrypted data file and shares on the blockchain (currently implemented using a smart contract).

Share Generation Output

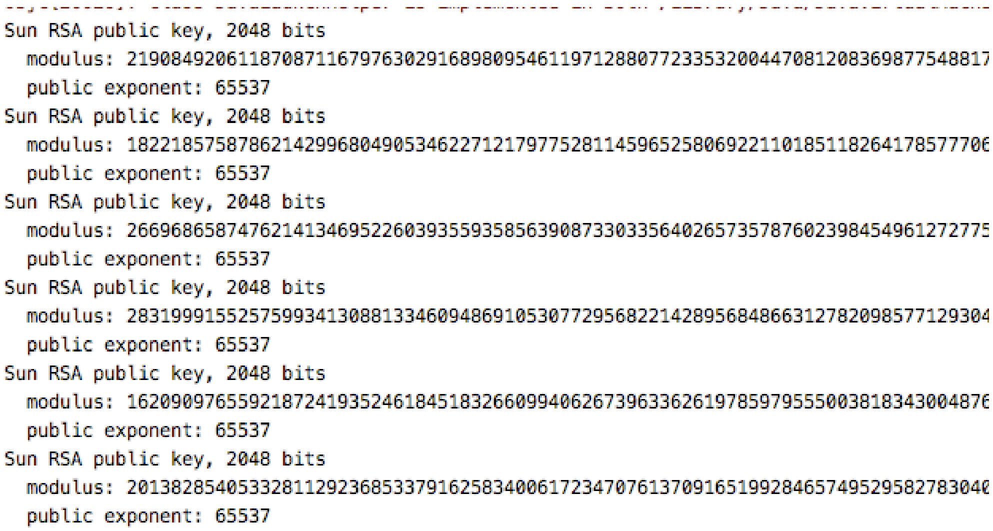
- A screenshot of the secret sharing map printed (after execution) is shown below. We map index to polynomial evaluated at that index:



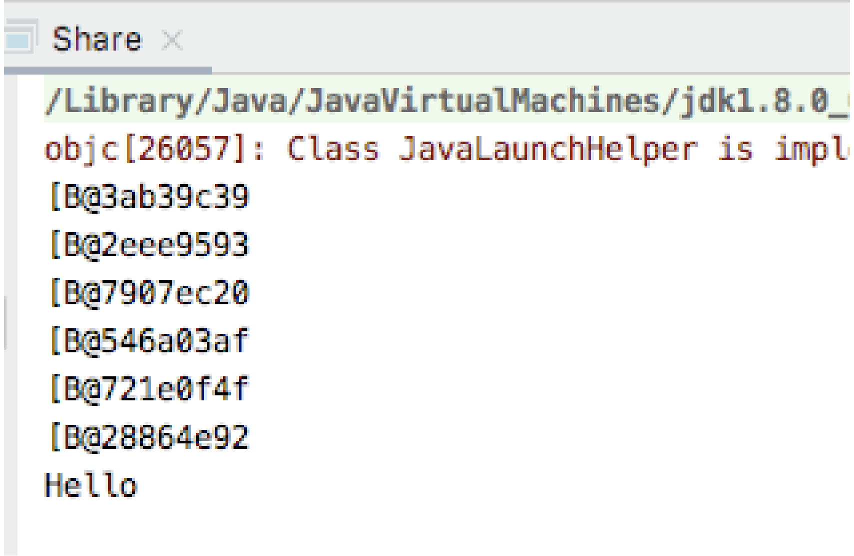
```
/Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/
objc[25909]: Class JavaLaunchHelper is implemented in both /Libr
0
[B@2aafb23c
1
[B@2b80d80f
2
[B@3ab39c39
3
[B@2eee9593
4
[B@7907ec20
5
[B@546a03af
Hello

Process finished with exit code 0
```

Encrypting Miner shares with Public Keys

We encrypt the miner public keys using RSA encryption scheme. A screenshot of the public keys and encrypted shares when the code is executed is shown below (Where n=6):



```
Sun RSA public key, 2048 bits
  modulus: 219084920611870871167976302916898095461197128807723353200447081208369877548817
  public exponent: 65537
Sun RSA public key, 2048 bits
  modulus: 182218575878621429968049053462271217977528114596525806922110185118264178577706
  public exponent: 65537
Sun RSA public key, 2048 bits
  modulus: 266968658747621413469522603935593585639087330335640265735787602398454961272775
  public exponent: 65537
Sun RSA public key, 2048 bits
  modulus: 283199915525759934130881334609486910530772956822142895684866312782098577129304
  public exponent: 65537
Sun RSA public key, 2048 bits
  modulus: 162090976559218724193524618451832660994062673963362619785979555003818343004876
  public exponent: 65537
Sun RSA public key, 2048 bits
  modulus: 201382854053328112923685337916258340061723470761370916519928465749529582783040
  public exponent: 65537
```

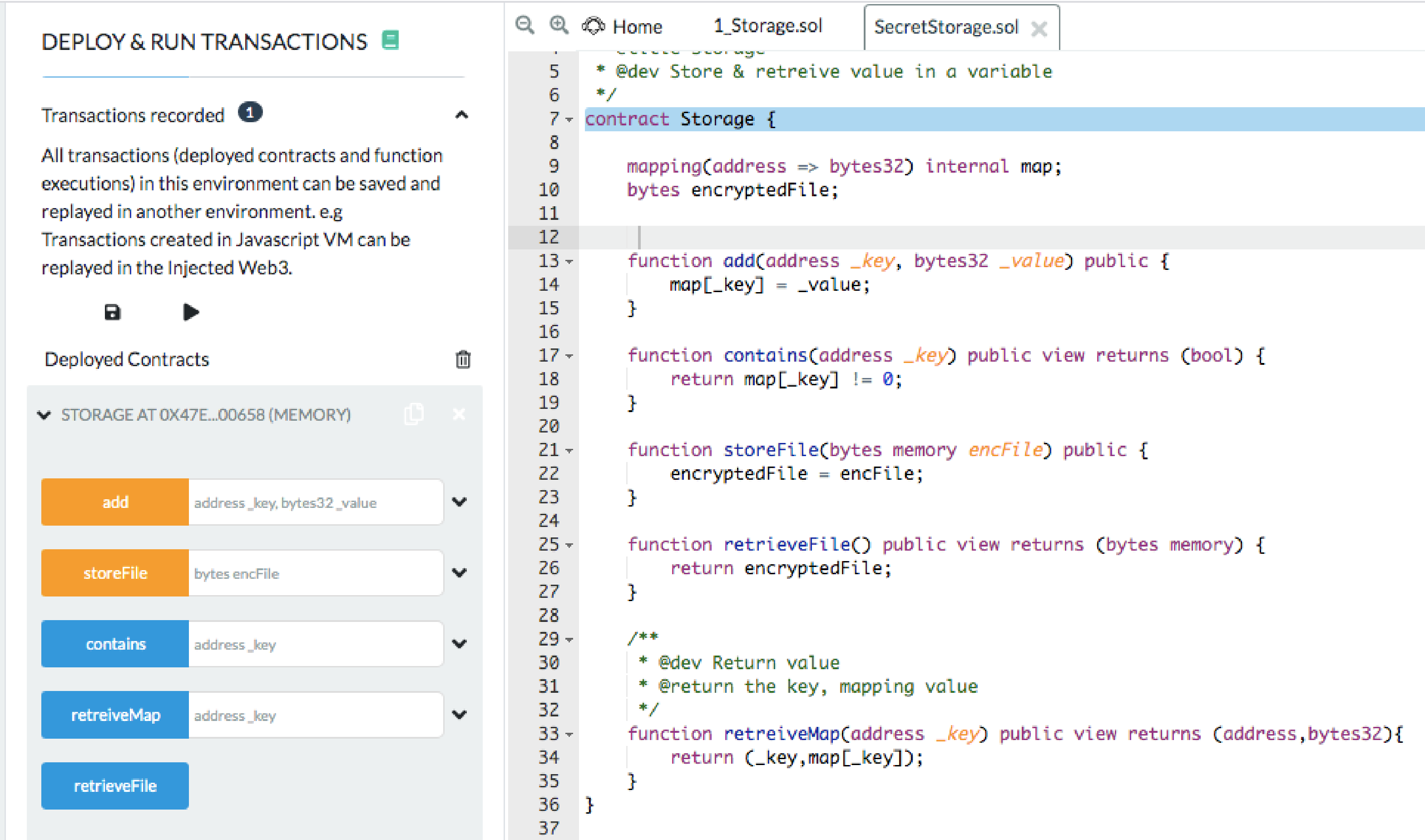


```
/Library/Java/JavaVirtualMachines/jdk1.8.0_
objc[26057]: Class JavaLaunchHelper is impl
[B@3ab39c39
[B@2eee9593
[B@7907ec20
[B@546a03af
[B@721e0f4f
[B@28864e92
Hello
```

Encrypting Miner shares with Public Keys

- We create a smart contract which stores a mapping from miner address to secret key share (of type bytes) with the following functions:
  - Add a share to the map
  - Store the encrypted file
  - Retrieve the share of a given miner address
  - Check if an address is in the map

Smart Contract:



DEPLOY & RUN TRANSACTIONS

Transactions recorded 1

All transactions (deployed contracts and function executions) in this environment can be saved and replayed in another environment. e.g Transactions created in Javascript VM can be replayed in the Injected Web3.

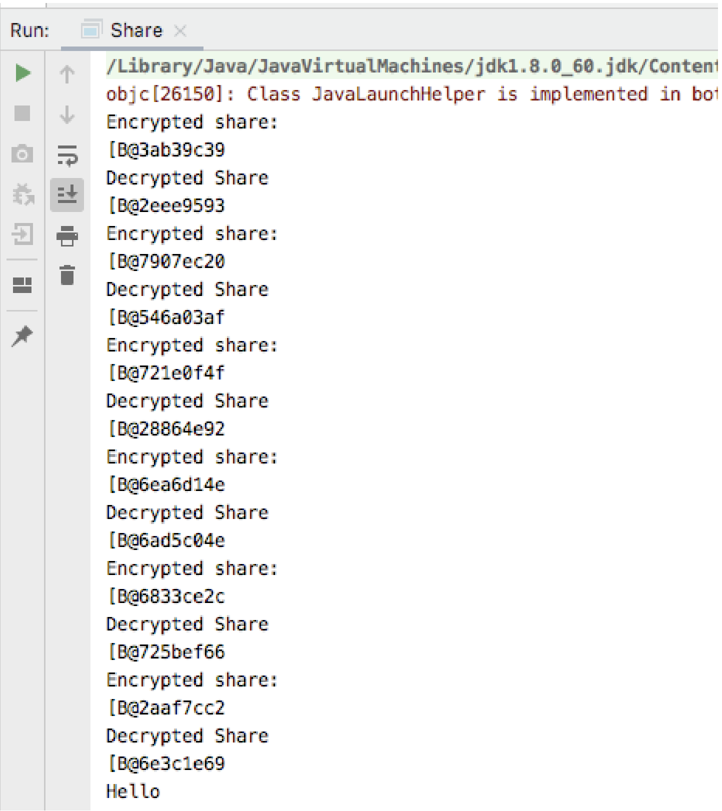
Deployed Contracts

```solidity
 * @dev Store & retreive value in a variable
 */
contract Storage {

    mapping(address => bytes32) internal map;
    bytes encryptedFile;


    function add(address _key, bytes32 _value) public {
        map[_key] = _value;
    }

    function contains(address _key) public view returns (bool) {
        return map[_key] != 0;
    }

    function storeFile(bytes memory encFile) public {
        encryptedFile = encFile;
    }

    function retrieveFile() public view returns (bytes memory) {
        return encryptedFile;
    }

    /**
     * @dev Return value
     * @return the key, mapping value
     */
    function retreiveMap(address _key) public view returns (address,bytes32){
        return (_key,map[_key]);
    }
}
```

- We use the RemixIDE to test the smart contract. We are able to run these functions implemented in the smart contract using the user interface on the left of the picture.

Miner Share Decryption

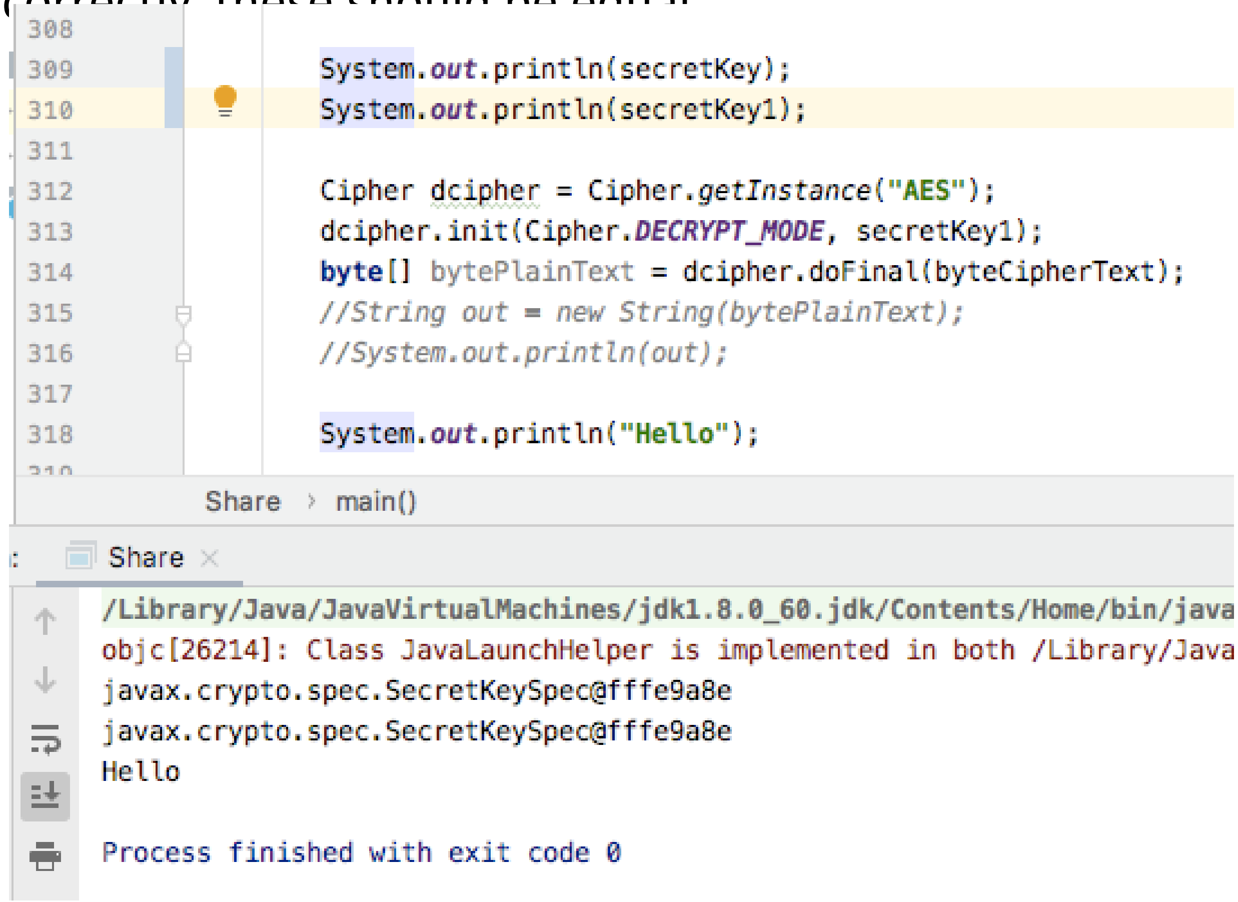
- Once miners take their shares from the blockchain, they are able to decrypt them using their private key. A picture of this code execution is shown below, with the encrypted share and then original share for each miner. In reality, each miner will only have to do this for their own share, but we implement for all for testing purposes.



```
/Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Conten
objc[26150]: Class JavaLaunchHelper is implemented in bo
Encrypted share:
[B@3ab39c39
Decrypted Share
[B@2eee9593
Encrypted share:
[B@7907ec20
Decrypted Share
[B@546a03af
Encrypted share:
[B@721e0f4f
Decrypted Share
[B@28864e92
Encrypted share:
[B@6ea6d14e
Decrypted Share
[B@6ad5c04e
Encrypted share:
[B@6833ce2c
Decrypted Share
[B@725bef66
Encrypted share:
[B@2aaf7cc2
Decrypted Share
[B@6e3c1e69
Hello
```

Recovering the Secret Key

- We then use this these decrypted miner shares to recover the secret key. Proof of working program is shown below, where we first print the original secret key (secretKey) and then the reconstructed secret key (secretKey1). If the program behaves correctly, these should be equal.

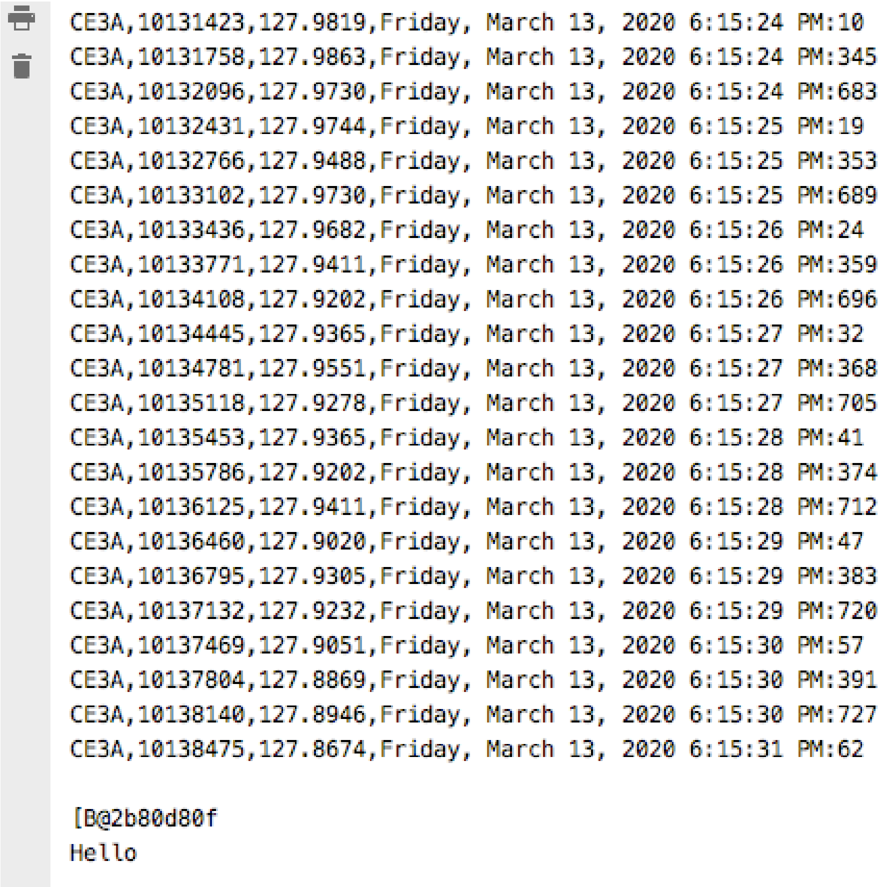


```java

        System.out.println(secretKey);
        System.out.println(secretKey1);

        Cipher dcipher = Cipher.getInstance("AES");
        dcipher.init(Cipher.DECRYPT_MODE, secretKey1);
        byte[] bytePlainText = dcipher.doFinal(byteCipherText);
        //String out = new String(bytePlainText);
        //System.out.println(out);

        System.out.println("Hello");
```

```
/Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/java
objc[26214]: Class JavaLaunchHelper is implemented in both /Library/Java
javax.crypto.spec.SecretKeySpec@fffe9a8e
javax.crypto.spec.SecretKeySpec@fffe9a8e
Hello

Process finished with exit code 0
```

Decrypting ciphertext to retrieve private data

- With the secret key recovered, we are able to then decrypt the data and recover the original CSV file. A picture of the code execution is shown below. We first print the decrypted file (CSV) and then the encrypted byte[] version.



```
CE3A,10131423,127.9819,Friday, March 13, 2020 6:15:24 PM:10
CE3A,10131758,127.9863,Friday, March 13, 2020 6:15:24 PM:345
CE3A,10132096,127.9730,Friday, March 13, 2020 6:15:24 PM:683
CE3A,10132431,127.9744,Friday, March 13, 2020 6:15:25 PM:19
CE3A,10132766,127.9488,Friday, March 13, 2020 6:15:25 PM:353
CE3A,10133102,127.9730,Friday, March 13, 2020 6:15:25 PM:689
CE3A,10133436,127.9682,Friday, March 13, 2020 6:15:26 PM:24
CE3A,10133771,127.9411,Friday, March 13, 2020 6:15:26 PM:359
CE3A,10134108,127.9202,Friday, March 13, 2020 6:15:26 PM:696
CE3A,10134445,127.9365,Friday, March 13, 2020 6:15:27 PM:32
CE3A,10134781,127.9551,Friday, March 13, 2020 6:15:27 PM:368
CE3A,10135118,127.9278,Friday, March 13, 2020 6:15:27 PM:705
CE3A,10135453,127.9365,Friday, March 13, 2020 6:15:28 PM:41
CE3A,10135786,127.9202,Friday, March 13, 2020 6:15:28 PM:374
CE3A,10136125,127.9411,Friday, March 13, 2020 6:15:28 PM:712
CE3A,10136460,127.9020,Friday, March 13, 2020 6:15:29 PM:47
CE3A,10136795,127.9305,Friday, March 13, 2020 6:15:29 PM:383
CE3A,10137132,127.9232,Friday, March 13, 2020 6:15:29 PM:720
CE3A,10137469,127.9051,Friday, March 13, 2020 6:15:30 PM:57
CE3A,10137804,127.8869,Friday, March 13, 2020 6:15:30 PM:391
CE3A,10138140,127.8946,Friday, March 13, 2020 6:15:30 PM:727
CE3A,10138475,127.8674,Friday, March 13, 2020 6:15:31 PM:62

[B@2b80d80f
Hello
```

Work to be done:

- Create smart contract for some subset of miners to post their decrypted shares (which can then be taken by any miner for secret key reconstruction)
- Improve the user interface for adding shares and encrypted file in the smart contract (currently run in RemixIDE), and share generation, file encryption (currently run in the command line)
- Simulated cyberattacks for system robustness
- Will be updated in GitHub repository

Deliverables and Timelines

Task / Subtask Number	Deliverable Title	Due Date
1.0	Project Management Plan	Update due 30 days after award. Revisions to the PMP shall be submitted as requested by the NETL Project Manager.
2.0	Sensor Networks for Fossil Power Generation System	Delivery to NETL 6 months after the start of the project.
3.0	Secure transmission of sensors to blockchain nodes	Delivery to NETL 3 months after Task-2.0, i.e., 9 months after the start of the project.
4.0	Smart Private Ledger Blockchain (codes and algorithms)	Delivery to NETL 12 months after the start of the project.
5.0	Hierarchical Access Control for Data Retrieval (codes and algorithms)	Delivery to NETL 3 months after the Task-4.0, i.e., 15 months after the start of the project
6.0	Robust Blockchain Including Necessary Modifications Ready to be Implemented in the Field	Delivery to NETL 9 months after the Task-5.0, i.e.,24 months after the start of the project

Challenges and Risks

No	Risks	Probability	Impact	Mitigation
i.	Delay in the formation of sensor networks: The PIs propose to create high temperature sensor networks at CMU by leveraging a prior NETL project on sensors and using aerosol jet printing technology. There is a risk for equipment breakdown and the sensor networks not being ready by the end of the third quarter	Low	High	1. Warranties/service agreements with the manufacturers are in place for the equipment. 2. The PIs will use individual commercial temperature sensors in case the sensor network fabrication is delayed.
ii.	Risk for wireless transmission: There is a low probability that the sensor networks cannot send the signal wirelessly to the blockchain nodes.	Low	Moderate	1. The PIs will use commercial wireless sensors (two) as a back-up to demonstrate the concept 2. Multiple suppliers are available in the market with wireless sensors and will be utilized as necessary.
iii.	Risk for formation of Blockchains: there is a small probability that the continuous stream of data coming from sensor readings will cause scalability issues in the blockchain	Low	Moderate	1. The PIs will increase the block size to handle a larger number of transactions per second 2. The number of new blocks per unit time could also be increased to improve the scalability of the system
iv.	Risk for data retrieval: there is a risk that if a number of nodes on the Blockchain go offline, the data stored could become inaccessible	Low	Moderate	1. This risk can be mitigated by increasing the number of nodes. The higher the number of nodes, the better the availability of the system would be. In any case, compared to a centralized data storage, the system will provide much higher level of anonymity.

Acknowledgements

- Robie Lewis, Dr. Vito Cedro, and Dr. Sydni Credle for help on guidance of the project

Acknowledgement and Disclaimer

Acknowledgment: "This material is based upon work supported by the Department of Energy Award Number DE-FE0031770."

Disclaimer: "*This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.*"

Questions?