**ECE463/464 Capstone Final Project Report**

*Cyber Defense of Robust Microgrid Power Systems*

**C1C Brett Martin, C1C Dustin Priaulx, C1C Julian Flores**

Prepared for

**Project Mentor Dr. Glen Dudevoir, Course Director Dr. John Ciezki,**

**ECE Department Head Col. Brian Neff**

**March 4, 2020**

# Abstract

In the operational Air Force, cyber threats pose a serious threat to the continuity of operations and the integrity of our information. One significant aspect of our infrastructure that demands a robust line of defense against these threats is power systems. Without proper security against enemy actors in both the physical and cyber domains, nearly every facet of Air Force operations can be brought to a grinding halt. Events involving attacks on power systems have already occurred in other parts of the world such as the cyber attack that crippled three Ukraine’s power control systems in 2015.

The goal of the U.S. Air Force Academy Cyber Power capstone team is to provide solutions to such vulnerabilities across several points of entry and against various means of exploitation. To this end, solutions must involve hardening physical defenses within our own microgrid as well as providing integrity for the data being transmitted between networked nodes.

To accomplish these tasks, certain aspects of the project are split up among the team. Our team is currently split among three different tasks. The first team is creating a variable PQ load, which will give our testbed the capability to alter the amount of real and reactive power that is consumed at the Industrial Area. Our second task is to create a robust protection scheme within the SEL relays to quickly identify and trip the appropriate breakers, and then reroute power through normally unenergized transmission lines, providing redundant and resilient avenues of power. Lastly, a variation of Swirld’s Hashgraph algorithm will be implemented on each relay within the microgrid to ensure that the data being received by the Real-Time Automation Controller (RTAC) is accurate. By using Byzantine Fault Tolerance, the Hashgraph algorithm will provide a fast and reliable means of automatically correcting fault or tampered-with data across the entire testbed and report any anomalies to a central administrator console.

Table of Contents

[Abstract 4](#_Toc11497613)

[1 Project Description 6](#_Toc11497614)

[2 Use Case 6](#_Toc11497615)

[3 System-Level Requirements 7](#_Toc11497616)

[4 Functional Architecture 7](#_Toc11497617)

[5 System-Level Trade Studies 9](#_Toc11497618)

[6 Cyber-Physical Architecture 11](#_Toc11497619)

[7 Subsystem Descriptions 12](#_Toc11497620)

[7.1 Subsystem 1 12](#_Toc11497621)

[7.1.2 Modeling/Analysis/Simulation/Calculations 12](#_Toc11497622)

[7.1.3 Circuit and/or software and/or mechanical hardware diagram 12](#_Toc11497623)

[7.1.4 Testing 12](#_Toc11497624)

[7.2 Subsystem 2 13](#_Toc11497625)

[7.3 Subsystem 3 13](#_Toc11497626)

[8 System Integration 13](#_Toc11497627)

[8.1 Procedures 13](#_Toc11497628)

[8.2 Results 13](#_Toc11497629)

[8.3 System Validation 13](#_Toc11497630)

[9 Evaluation 13](#_Toc11497631)

[9.1 Strong points of your design 13](#_Toc11497632)

[9.2 Weak points of your design 13](#_Toc11497633)

[10 Project Management 14](#_Toc11497634)

[10.1 Work Breakdown Structure 14](#_Toc11497635)

[10.2 Schedule 14](#_Toc11497636)

[10.3 Budget(s) 15](#_Toc11497637)

[10.4 Risk Management 17](#_Toc11497638)

[10.5 Configuration Management 18](#_Toc11497639)

[11 Conclusions 18](#_Toc11497640)

[11.1 Lessons learned 18](#_Toc11497641)

[11.2 Future work 18](#_Toc11497642)

[12 References 19](#_Toc11497643)

[Appendices 20](#_Toc11497644)

# 1 Project Description

The Cyber Power Capstone Project is an Industrial Control System (ICS) that emulates the USAFA power grid. The goal is to provide a means to holistically protect the power grid system from both physical and cyber threats to ensure a more robust, resilient, and flexible system overall. The hardware must be made safer and more reliable for consumers while the software must be hardened to ensure data integrity.

In recent history, attacks on ICS’s and Supervisory Control and Data Acquisition (SCADA) systems have become a significant threat. It was once believed that control systems operating on proprietary software were in no way susceptible to exploitation because of the unique nature of such software environments. Attacks such as Stuxnet—a computer worm that was discovered as an attack on Iranian uranium centrifuges—revealed to the world that such attacks were possible and, when executed properly, serve as a uniquely effective tool against ICS’s (Anderson).

Years later, in 2015, an attack on several Ukrainian power systems revealed to the world that power substations were no different from any other ICS and could be exploited. These attacks caused power outages spanning several cities and lasting for several hours. If such an attack were to occur in a U.S. area of responsibility (AOR), the cascading effects could be devastating to military and strategic operations globally.

In previous capstone efforts, several areas of concern related to the security of the testbed had been explored and built on. These efforts include creating robust protection schemes for the test bed through programming the relays. Unfortunately, these settings since have been overwritten by more-less many default settings. A few lines of protection logic were salvaged, but the rest of the protection settings are in the process of being restored. The team is creating ample documentation and backup files in order to prevent something like this from happening again. Other previous Capstone efforts focused on many hardware components that would add capability to our test bed. Some hardware components include a Resistive Load Bank capable of providing up to a simulated ~15MW of consumption and a no-longer-working Fault Generator, supposed to be able to simulate 0Ω line to line and line to neutral faults, both controlled by Python scripts on Rasberry Pi’s. The last piece of hardware last year’s team worked on was the simulation of man-in-the-middle attacks over ethernet via an FPGA board.

Though the FPGA efforts had produced no significant deliverable in previous years, an alternative to simulating man-in-the-middle attacks is to perform such a simulation in software. With the use of Raspberry Pi computers affixed to each relay, an alternative to the Hashgraph data structure and algorithm will be employed on each of these computers. In doing so, a Byzantine Fault Tolerance method of fault detection—in which it is assumed that no more than one-third of nodes on a network can be malicious—and automatic correction of faulty data can be used to ensure that man-in-the-middle attacks and other anomalies in the testbed are mitigated.

# 2 Use Case

This use case will demonstrate what the test bed system should be able to do if the fault is introduced in the system. Another use case will demonstrate what the system should be able to do in the case of a Man In the Middle Attack that is spoofing bad data.

Figure 2.1 shows the normal configuration of the test bed. All lines and breakers that are red mean that those lines and breakers are energized and are allowing current through them. All lines and breakers that are green are open and un-energized. In this nominal configuration, Transmission Lines (TL) 2 and 3 are usually open, only closing if there is a fault on TL4 or TL1, respectively.

Let us look at a specific case for a line to ground fault on TL4, shown illustratively on Figure 2.2. A Line-to-Ground fault will introduce an increase in current on TL4. Relay 421-D on the bottom side of CSU2 has a sensor that will detect the increase in current, compare it against its internal trip logic, and then trip if the current is too high. For this demonstration, the current will exceed the maximum allowed current setting on the relay. In response, this relay will open breaker C25, and the send an SEL Remote Bit through serial to relay 487-B, signaling that relay to trip Breaker C52. Both breakers will attempt to reclose three times, seeing if the fault is cleared. If it is not, both breakers will remain open and isolate the fault on TL4. This leaves the South Substation without power. To reroute power to it, the relay 487-B will close breaker C51 and then send a SEL Remote Bit to Relay 421-B, the relay controlling Breaker C15 on the bottom side of CSU1. Relay 421-B will receive this mirrored bit and in turn close Breaker C15. This action will then energize TL2, which would then provide the alternate avenue of power to re-energize the South Substation and restore power back to the Industrial Area.

In addition, the Hashgraph algorithm works as follows. Originally designed as a ledger technology for cryptocurrency applications, the modified version will be repurposed to keep track of sampled data from relays within the test bed. In the original version, “events” contain transactions, the time in which the transaction took place, and a hash of the current event and the previous event. Additionally, “members” in the original algorithm are analogous to nodes or relays within the test bed in the modified version.

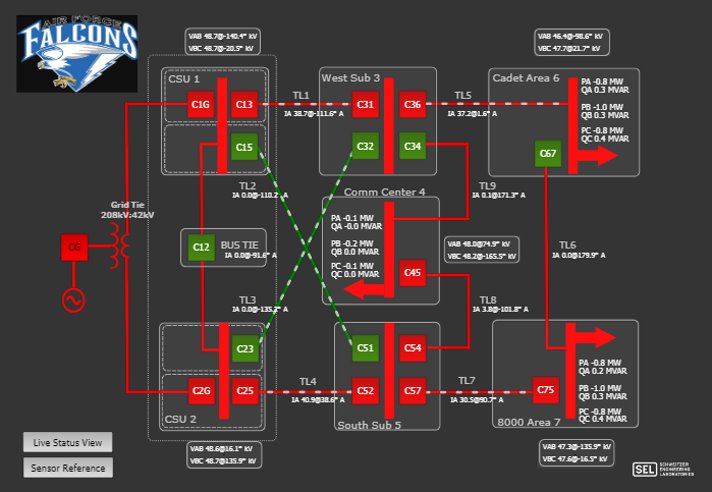
The algorithm is run on each computer in the network and communication between these nodes are established. With each new connection, a “member” is added to the Hashgraph, which is a directed acyclic graph that contains the members, “events” for each member, and the connections representing each instance of a member sending event details to another member. This data structure can be seen in Figure 2.2 as a time-driven series of samples over time and the routes in which sample data takes when being distributed among the other members.

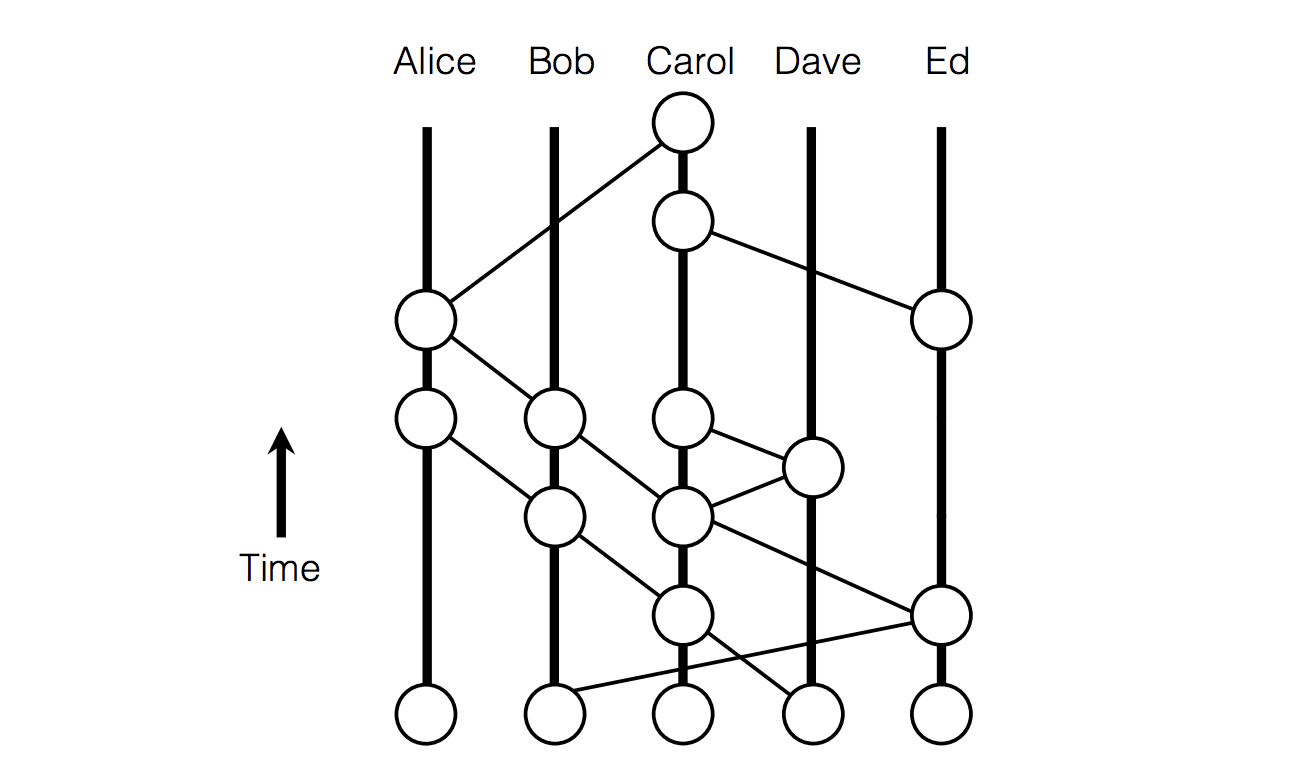
When a sample is taken by a node, or a member, that sample data is sent to two random nodes. Those two nodes send that sample data to two more random nodes. This continues until there are no more nodes left in which to distribute data. This procedure is called “gossping.” The nodes then come to a consensus as to what the appropriate sample data value is. By using Byzantine Fault Tolerance, which assumes that no more than one-third of the network are malicious actors, the majority vote will be the accepted value. This will also be useful for detecting faults, anomalies, or potential attacks. Additionally, by hashing the events prior to sending them, the receiving node will be able to tell if the data has been tampered with through use of PKI.

If bad data is introduced into the network—either by means of a fault or a malicious actor—then the algorithm, by means of the methods mentioned above, will automatically correct the data for all nodes. By coming to a consensus as to which nodes have the correct data, the ones with improper sample data can recreate their Hashgraph struct based on that of the correct nodes. This will also queue the administrator console into the fact that an anomaly has occurred. Further research will determine whether or not the exact location of the threat or fault can be determined.

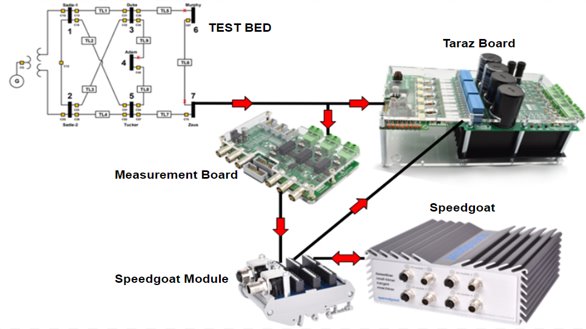
Another use case demonstrates the capabilities of adding the adjustable PQ Load. The PQ Load uses three-phase currents and voltages that are inputted into the three-phase voltage to DC voltage rectifier which is then inputted into a buck chopper that steps the voltage down and dissipates the power through a resistance. Measurements are also taken from bus 7 in order to get real time data on current and voltage into the SpeedGoat. The SpeedGoat uses Simulink logic to implement consistent current and voltage for the rectifier and buck chopper. Furthermore, the SpeedGoat controls the gates on the rectifier to allow for phase lock loop in order to change the inputted current to manipulate the real and reactive power consumed by the load. Figure 2.3 shows a diagram of where the 3 phase voltages and current are being sent. The adjustable real and reactive power load will be a key piece to implement into our grid.

The PQ Load is essential in order to add more functionality to our grid. Currently, the make shift resistive load that is connected to bus 7, the industrial area, only consumes about 1 MW of power. The PQ Load will be able to consume up to 15 MW of real and reactive power per phase. The PQ Load shall have the capability to create a balanced and an unbalanced load in order to test the resiliency of our grid. If and unbalanced load is at the industrial area of our grid, it is important to see how our system responds to the shift in current lag time. There are settings on the relays that can cause a relay to trip based on an unstable phase difference.

**Figure 2.1: HMI for Cyber Power Testbed**



**Figure 2.2: Basic Hashgraph data structure design**



**Figure 2.3: Diagram of PQ Load**

# 3 System-Level Requirements

There are 6 key requirements for the Cyber Power capstone team. When the project was first started, most of the components that were supposed to work, no longer worked. The testbed was missing the integration of a safe protection scheme for the SEL relays to detect faults and close breakers on these detections. Furthermore, the system was not able to reroute power in the case of the detection of a fault. The system shall be able to reroute power in the case of a fault on a transmission line in order to keep power running on the base. The point of the redundancy in the power grid created is to supply power through different transmission lines in the case of a fault or maintenance. To further protect the system, the system shall be able to detect when information being received from a relay is incorrect and will automatically correct it. Using a HashGraph algorithm, the data will be checked to see if it has been manipulated by a “man-in-the-middle attack.” This can potentially be dangerous because a hacker could potentially tap into the SEL communications to change data. The hacker could potentially compose a plan to create a line to neutral fault that could destroy generators, transmission lines, and transformers on the line which the hacker could potentially cover up by sending spoofed safe data to the relay that was supposed to trip. Furthermore, the Cyber Power capstone team will continue the development of the PQ Adjustable Load. The PQ Load is important to have in the system in order to simulate an industrial load which may be consuming both real and reactive power. This is realistic because the motors used at industrial plants also consume reactive power. Having a load to simulate real and reactive power will help develop and test settings on the relays in order to allow for a more resilient grid. The PQ Adjustable load has some cleanliness problems. Once the PQ Load is working, measures shall be taken to address the cluttered construction of the PQ Load in order to create a more compact and consolidated load that can fit in the testbed – allowing for a more aesthetically pleasing design. Finally, the testbed shall be able to transmit data using high frequency radio. This will increase the testbed’s capabilities.

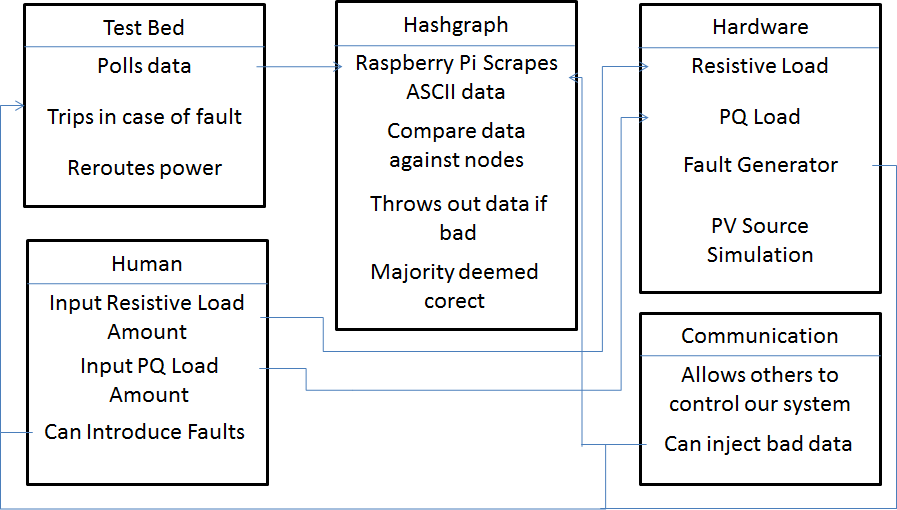
Below is a consolidated Requirement Traceability Matrix that shows six key requirements for our test bed system, describing the requirement, our deliverable, and why each one is an important key requirement. Requirements 1 & 2 are our most pertinent requirements because Requirements 4 and 5 depend on them to work. Requirement three is able to be worked on independently of 1 & 2, and Requirement 6 is a non-pertinent requirement and will be worked on most likely post-lesson 30.

**Table 3.1: Key Project Requirements**

|  |  |
| --- | --- |
| 1 | **Statement:** The Protection scheme in place must be able to trip breakers in the case of a fault proportional to the current reading. |
| **Derived from User Need (if applicable): N/A** |
| **Verification Method:** Through testing, demonstrate that the system is able to detect a fault and trip the appropriate breakers under the correct conditions. |
| **Why is This a Key Requirement:** This is a key requirement because our power grid must be protected from faults to prevent structural damage |
| **Time:** 18 lessons |
| **Cost:** $0.00 Already have all infrastructure |
| **Risk:** Programming BKRs Trip Logic is a very specialized task that requires critical aggregation of data and knowledge of current protection in other breakers. |
| 2 | **Statement:** The protection scheme must be able to reroute power in the case of a fault within 10 seconds. |
| **Derived from User Need (if applicable): N/A** |
| **Verification Method:** Through visual demonstration, show that the system accurately reroutes power in the case of a fault. |
| **Why is This a Key Requirement:** This is a key requirement because the relays should automatically be able to communicate with each other to reroute power in the case of a fault to continue supplying power to as many customers as possible |
| **Time:** 18 Lessons |
| **Cost:** $0.00 Already have all infrastructure |
| **Risk:** Programming BKRs Trip Logic is a very specialized task that requires critical aggregation of data and knowledge of current protection in other breakers. |
| 3 | **Statement:** The system shall be able to detect when information being received from a relay is incorrect and will automatically correct it. |
| **Derived from User Need (if applicable): N/A** |
| **Verification Method:** Create system of RasPi that are able to scrape ASCII Data off relays. Hashgraph algorithm will provide verification method. |
| **Why is This a Key Requirement:** Incidents like the Ukraine Power Station exploits in 2015 and Stuxnet have proven that industrial control systems require strict security and data integrity to prevent serious system failures or malfunctions resulting in detrimental losses to the overall Air Force mission. |
| **Time:** 20 – 30 Lessons, time independent of 1 and 2 |
| **Cost:** $1030 (10 RasPi + all adapters and wires) |
| **Risk:** Insufficient knowledge of networking protocols and data aggregation from Relays via Telnet. |
| 4 | **Statement:** The load must be able to simulate different amounts of real and reactive power. |
| **Derived from User Need (if applicable): N/A** |
| **Verification Method:** Demonstrate that the PQ load is able to generate different amounts of real/reactive power. |
| **Why is This a Key Requirement:** Loads on the power grid are rarely 100% resistive. We need to have this PQ load in order to accurately simulate the loads that consume reactive power on the power grid. |
| **Time:** 7 Lessons, 2 to hook up PQ load to Bus 7, 5 to implement Speedgoat module using Simulink Logic |
| **Cost:** $0.00 |
| **Risk:** Damaging current set up if initial current spike to bring PQ load online is dangerously high |
| 5 | **Statement:** PQ Load will be clean and compact enough to be able to mount onto the test bed. |
| **Derived from User Need (if applicable): N/A** |
| **Verification Method:** Demonstrate that the PQ load is able to generate real and reactive power by measuring currents. |
| **Why is This a Key Requirement:** This is a key requirement because adding a more realistic load that consumes reactive power will test our power grid’s resiliency. This is mostly a cosmetic step. |
| **Time:** Dependent on functionality of the PQ Load. |
| **Cost:** At most $200 for PCBs, banded wires, etc. |
| **Risk:** A risk with cleaning up the PQ Load would be to not correctly document wiring. This could cause the real and reactive load to be wired incorrectly and therefore no longer work. |
| 6 | **Statement:** The testbed must be able to communicate with other testbeds in the case of an internet outage. |
| **Derived from User Need (if applicable): N/A** |
| **Verification Method:** (Tentative) Transmit data to other testbeds through internet and radio. |
| **Why is This a Key Requirement:** This is a requirement in order to allow 3rd parties the opportunity to hack into our testbed and test our cyber security defenses |
| **Time:** Unknown lead time for grant/equipment, dependent on a fully functional testbed |
| **Cost:** $7,500 of equipment, pending $10,000 NSA grant |
| **Risk:** Might not have the money to acquire materials to work on this requirement until late in the year. |

4 Functional Architecture

The system will be split up into five major components. The first major component is deals with the relay settings and protection schemes themselves. The second one deals with the hardware of the system, specifically the resistive load, the PQ load, the fault generator, and eventually, the PV source simulator. The third component deals with data integrity through the hashgraph algorithm. The fourth aspect deals with the human aspect, where the user will be able to input the amount on the resistive load and the amount of PQ on the PQ load via a RasPi controlled GUI. The last aspect of our project will deal with communicating with other remote testbeds that are geographically separated from ours. Figure 4.1 shows the types of interdependencies that may exist between components. Arrows exiting blocks show outputs, which are then inputs to the way the arrow is pointing. Some arrows merge with other arrows, indicating that any input can be satisfied for that specific aspect to work. For example, “Trips in case of fault” can work if provided by either fault generator.

.

**Figure 4.1: Cyber Power Project Functional Architecture**

The Hashgraph algorithm will be implemented on a series of Raspberry Pis. Each computer will be connected to a relay via an ethernet cable so that they can be connected to via Telnet over port 23 using an application such as PuTTY. Each computer will be connected to the wireless router to simulate communication over a Wide Area Network (WAN). Each computer will also have a connection to a KVM through which all computers in the Hashgraph network will be accessible.

The algorithm will be written in Python because of its ease of programmability and the abundance of Raspberry-Pi-related libraries and features. The first step in the algorithm is to generate a key pair that will be used to hash the events that will be sent to other nodes. The algorithm will then establish a connection to the relay and verify that sample data can be obtained. Connections to the other relay computers will be established and the Hashgraph will be populated with member nodes until the prescribed number of members are all connected.

Once connections between relays has been established, the computers will begin collecting sample data from the relays and creating event objects containing the sample data, sample time, member data—to include node name and public key—and the hashes of the event and the event prior (except for the first event). Hashing the event will give the receiving computer a secondary means of determining whether the information is reliable or not. The algorithm will wait until all nodes in the network have received the data and the consensus can begin. When over two-thirds of the network have the correct data, all other nodes with incorrect data are overwritten with the correct Hashgraph that was agreed upon in the consensus and the outliers are reported to the administrator console.