References

**Brewster, C., Roussaki, I., Kalatzis, N., Doolin, K., & Ellis, K. (2017). IoT in agriculture: Designing a europe-wide large-scale pilot.*IEEE Communications Magazine, 55*(9), 26–33.**

IOT is useful in Agriculture and other means. This paper recommends a large scale pilot be used that is funded by the government to figure out if this system works. It brings in many problems associated with IOT.

“There are a great variety of different types of actors in the food system ranging from very large to verty small. Consequently no single solution, whether technological, business model, or regulatory will fit or accommodate the needs of all.”

Interoperability is also a concern so there should be a way to address this issue. “Thus, a key requireemnet here is that all systems provide export facilities or API access that return standard formats, typically XML or JSON, and where possible legacy systems are provided with appropriate interchange gateways.

**Ojha, T., Misra, S., Raghuwanshi, N. S., Ojha, T., Misra, S., & Raghuwanshi, N. S. (2015). Wireless sensor networks for agriculture: The state-of-the-art in practice and future challenges.*Computers and Electronics in Agriculture., 118*, 66–84. doi:10.1016/j.compag.2015.08.011**

**Bulut, C., & Wu, P. F. (2023). More than two decades of research on IoT in agriculture: A systematic literature review.*Internet Research, 34*(3), 994. doi:10.1108/intr-07-2022-0559**

Study on two decades of IoT in agriculture. Determined that “the “Big three” barriers for the overall sector are cost, skills, and standardization.” The majority “focus in the protected indoor environment.” “Iot” is poised to be the next big “thing” after the advent of computers, Internet, and smartphones, leading to the fourth industrial revolution.” It can help agriculture grow their yield, it gives economic benefit and can help reduce water consumption.

What are the main barriers? The consensus of scholars is that there is a high “cost of upfront investment, lack of robust connectivity, and high power consumption, with limited access to energy sources and short battery life.

There is a concern with a diversity of process for IoT solutions. There is also a concern with connectivity. \*(Can this be addressed with a central server?)

IoT overview shows a layers approach which is a great visual. It can be a 3-5 layer system with the device, network, and analytics, possibly adding on business data analysis.

So how can we overcome these barriers?

Devices- There are novel approaches using paper based sensors, can use solar powered (varandas et al. 2020)

Network 4G and 5G technology expensive and tough to use. LOrA is promising.

**Kassim, M. R. M. (2020). Iot applications in smart agriculture: Issues and challenges. Paper presented at the *2020 IEEE Conference on Open Systems (ICOS),*19–24.**

This goes into actual use of IOT devices including a set up for a mushroom greenhouse. It does mention some free open source farming software. It goes into the importance of weather monitoring, soil content monitoring, disease monitoring, and irrigation. These are the ways to use IOT in a green house environment. What is interesting is that it brought up controlling a device by the IOT. So these two way devices are useful for IOT applications as employees do not need to be in the space to control it. So if the soil moisture content is too low we can water it.

Issues and challenges include power consumption, harsh environments, networking, infrastructure, communication signal (rural areas), reliability, awareness, and security. Many of these concerns are written in a broad sense in this paper and not detailing ways to fix or exact issues.

**Kour, V. P., & Arora, S. (2020). Recent developments of the internet of things in agriculture: A survey.*Ieee Access, 8*, 129924–129957.**

Extensive study on IOT devices. In particular this paper goes over the history of agriculture and IOT devices. “Agriculture is growing from precision to micro-faming.” With that transition we have to stay on top of available technology to assist farmers. The author presents their current vision of the layers of an IOT device. This model was then updated and simplified. One thing that my device does is simplify this layer by utilizing internal software on a phone to save complexity that the farmer sees.

The paper then goes into each layer and what they entail including details on IOT devices, programming languages and network devices. Noting “NFC and Bluetooth are most suitable for greenhouses.” Which is an interesting takeaway and likely due to the range of the device. The author proceeds to discuss the different sensors and usage in agriculture and will be good to discuss in this paper as it covers which sensor package works for this device.

The literature shows that common parameters are “soil data, pH value, humidity level, moisture content, and water content.

The paper closes by covering the challenges and future work in IOT devices including the common trends of the cost of the device, standardization of the technology. There are interoperability issues with such a wide range of devices. The accessibility for developing nations for the infrastructure needed is difficult so a future study of a system with minimal infrastructure is important and can be discussed.

Overall “a design of platforms in a user-friendly manner” is key to making an iot device successful in the real world. Assuming the device is set up by anyone in an easy manner with a simple UI would be a huge help for agriculture types.

**Kuaban, G. S., Czekalski, P., Molua, E. L., & Grochla, K. (2019). An architectural framework proposal for IoT driven agriculture. Paper presented at the *Computer Networks: 26th International Conference, CN 2019, Kamień Śląski, Poland, June 25–27, 2019, Proceedings 26,*18–33.**

**Quy, V. K., Hau, N. V., Anh, D. V., Quy, N. M., Ban, N. T., Lanza, S., . . . Muzirafuti, A. (2022). IoT-enabled smart agriculture: Architecture, applications, and challenges.*Applied Sciences, 12*(7), 3396.**

This paper covers the usage of IOT in agriculture. “According to the Cisco forecast, over 500 billion IOT devices will be connected to the internet by 2030” “The primary motivation for their applications is the breakthrough progress of smart agriculture and its inevitable role as the future of smart and sustainable environmental management.”

IOT is broken down into IOT devices, Communication, and data processing. IOT device is the sensors needed for agriculture it includes Soil Moisture, Air Hum, Temp, UV, and actuators. Communications include short and long-range technologies, LoRA though “deploying a huge number of IOT devices for smart agriculture can cause interference to different network systems, especially sot IOT networks… including LoRA.”

The challenges of IOT include Economic efficiency which this paper provides an equation to show its worth. Which includes set up labor and maintenance labor. As well as the technical problems with interference, security, and reliability. Reliability is the durability of the system from damage through floods, rain, harsh environment etc.

**Wang, Q., & Yang, X. (2014). Research on IOT based special supply mode of agricultural products. Paper presented at the *2014 International Conference on Mechatronics, Electronic, Industrial and Control Engineering (MEIC-14),*1740–1743.**

**Varandas, L., Faria, J., Gaspar, P. D., & Aguiar, M. L. (2020). Low-cost IoT remote sensor mesh for large-scale orchard monitorization.*Journal of Sensor and Actuator Networks, 9*(3), 44.**

Similar to my design idea with a LoRa mesh. But why a router if we can use an android phone.

Study presents the design and construction of a low-cost iot sensor mesh that enables the remote measurement of parameters. The concern is that is costs $30 per transceiver and receiver so it adds up in the price. Though it is a very nice design.

\*My design would be a low-cost IOT sensor mesh utilizing Bluetooth LTE with a cellphone reader. The cellphone would route to the cloud. Making the Bluetooth LTE Inexpensive. Use is for more of a farm where the farmer would have to be in range to get the reading. (up to 100m).

Fighting against diseases is vital for famers. “Temperature and humidity are crucial to the surge of disease and pests, thus the development of a monitoring device to mitigate … is required.”

The device created uses a solar panel on a 5 device sensor that sends signal to a server via LoRaWAN. It is received via a LoraWan server that sends data to the cloud.

This is a great system with many advantages. The disadvantage being it is not receiving signals from the device. It is at a low speed. We can have a similar set up via BLE this will not be a consistent feed unless a cellphone is used as a simple router to upload to the cloud. Or we can use a device that just gets the sensor input when in the range such as at a greenhouse. Additionally this can be used for other applications.

Lora is placed in a sleep mode until an awake signal is received.

Maybe this device I make is more about the multi mode and less about agriculture as this seems to be the same style device?

**Kang Eun Jeon, She, J., Soonsawad, P., & Pai Chet Ng. (2018). BLE beacons for internet of things applications: Survey, challenges, and opportunities.*IEEE Internet of Things Journal, 5*(2), 811–828. doi:10.1109/JIOT.2017.2788449**

My paper can additionally delve into the battery saving characteristics of BLE. I need to figure out the range of the ble as well. Does the agricultural use need to have a longer range?

Paper:

BLE is good because of its low power consumption, compatible devices, and ease of integration. BLE can send more frequent data in short increments. IT is a great way to separate the devices as well in this use case ie. Temp, Pressure, Signals (on off ) etc. Additionally, it can be used as a one to many communication so anyone in the area can read it. It is famously used by apple in their location device.

This paper then shows what a payload looks like.

**yang ble ad hoc network paper**

This paper covers current and future usage of BLE technology. It goves over how the data payload works with the PDU's. This will be good to cover in the paper.

Based on our review of various ex-  
amples from emerging application domains, we further categorize  
four major and urgent challenges or additional requirements for  
BLE: 1) the need of enhanced data transmission, 2) mesh metworking  
support, 3) inter-operability with other wireless technologies, and  
4) more reliable privacy and security protection.

Range is a major drawback to using BLE but mesh networking can help. A paper was made to help with this.

**Low-Power BLE Relay Node Operation in Mesh-Like Architectures for Precision Agriculture -Gautam**

The article proposes a framework for precision agriculture that uses wireless sensor node clustering, but, instead of choosing cluster heads that aggregate and forward the data, battery-operated relay nodes are used whose sole purpose is to aggregate the data from the specific clusters assigned to them and forward the same.

Makes up a less frequent relay that can use batteries by making them learn the transmission pattern of the nodes so that they can be put to sleep in between receptions.

We propose that the low-power relay node “learns” and continuously adapts to the transmission schedule of each of the advertising devices (sensor nodes/ relay nodes) communicating with it, as a result of which it knows exactly when to wake up and receive the data from each node.

Maybe my design can show an alternative probabilistic approach to find and send ble data using relays and a low power mode that can meet a similar power consumption threshold without the need to “learn”. The authors procedure is to determine the system-up time which is the time elapsed since the relay node booted to getting the signal. This only scans between the sleep phase and the time it takes to boot. The issue is that maybe over time the boot phase can change depending on the device….

This is not a clock interval it simply can tell when the device typically sends and receives data and shuts off the nodes when data is not needed.

Another concern with this method is that the device can not send data randomly after a threshold is changed such as when the humidity in the soil gets too low it can not send a special detail….

**Probability**

The goal of the device is for the receiver to get the message every 5 minutes. There is no need for the receiver to send a signal back that needs a fast response. As such the relays and friends do not need to be constantly sending and receiving messages which will save time.

To create the model we will need to know what is the minimum number of blocks of time that the devices have to send (Msend) and listen (Mlisten) so that the message can be sent from one device to the next every 5 minutes with a high success rate. This success rate can change based on user as some might not care if they miss a meeting. This model can also change based on how often the end user needs to get the data.

The objective is to determine the probability that a receiver gets a message during the 5 minute window. The MSend value is chosen at random as well as the MListen value.

This is a low probability if we only listen during one block. But if the listener is taken over many blocks the probability of success increases. So we need to know using a repeated trial of n times where each trial is a success or failure. With that we can find how many times it will take for us to get a successful match. More simply what is the “probability of x successes in n trials for a binomial experiment”(Walpole, 144). The binomial probability mass function is the probability of ZERO success. What is the probability of zero success in 15 send tries in the probability success window of 7 listens in a total of 300 slots.

x = number of success overlaps

P = probability of success on a single trial

n = number of send tries

Now we can simplify this equation

Which can be simplified to:

The equations for combination is reduced to one

The probability of zero successes is:

The probability of success is then found by:

The probability of success on a single trial is:

For example we will assume that the number of successful tries n=300 and the number of listen windows of the Mlisten is 7.

It can be seen that with 7 random listen slots and 15 send tries we have a probability of success of 30% in 5 minutes where each send and receive takes one second. That is fairly good but it is not the optimized value.

To see what is happening this data is analyzed in R

#Create the number\_of\_send and number\_of\_Listen variables from 1 to 300

nSend<- seq(1, 300, by=1)

nListen<- seq(1,300, by=1)

#binomial distribution probability funciton

success\_binomial\_prob <- function(listens,blocks,sends){

p <- 1-((1-(listens/blocks))^sends)

return(p)

}

#first plot the succesful binomial probability with the 7 listens and 15 send blocks

results <- success\_binomial\_prob(7,300,15)

print(results)

print(results)

[1] 0.298228

Implementation

In reality the listen and send block length is not 1 second but should be 1 millisecond. As per research by XxXXXXX the send block and receive blocks are around 30ms. To implement this in the given equation the variables would change. The number of 1ms blocks in 5 minutes is 300,000.

To determine the effective slots which in the previous example was 300, the number of 30ms blocks have to be determined.

The new equation given the 7 random listen slots and 15 random receive slots is:

This is entirely too low of a probability. Therefore the number of send and receive blocks need to increase. As this is only 15 \* 30ms = 450ms. The device would only be on for half a second. To simplify the decision a 3d heat map can be used so that both changing variables can be adjusted as the x and y axis and the z axis can be the probability of success.

A graph with a curved line

Description automatically generated

A graph of a graph with numbers and lines

Description automatically generated

As can be seen in the plot 150 send slots and 150 listen slots gives a 89.6% probability of success. Given that each slot is 30ms the total time on for each device is 4500ms or 4.5 seconds out of 5 minutes. The amount of time that the device is on compared to the time slot is the duty cycle which can be used to show the usage of the device:

So the device will only be on for 1.5% of the time which should save a significant amount of energy for the relay devices.

**Simulation:**

To simulate these results 150 random send and receive slots need to be selected. From there an iteration can be conducted to see if both random values match in a total number of 10,000 slots. The simulation will run for 20,000 times to see if we get the same or similar probability to the calculation that was created:

#Simulation to see if the model is working properly

import random

#total number of slots and simulations

totalSlots = 10000

simulations = 20000

#chosen number of send and listen slots

send\_slots = 150

listen\_slots= 150

#initiate the number of successes

successes=0

#code needs to iterate over the total number of simulations and check if the send and listen slots overlap

for \_ in range(simulations):

#randomly select the send and listen slots

sent\_blocks = set(random.sample(range(totalSlots), send\_slots))

listen\_blocks = set(random.sample(range(totalSlots), listen\_slots))

#check if the send and listen slots overlap

if sent\_blocks & listen\_blocks:

#if they do, increment the number of successes

successes += 1

print(successes/simulations)

print(f"total time for send\_receive\_slots in 5 minutes: {(send\_slots\*30)/1000}s")

result:

> Success rate is : 89.785%

> Total time for send\_receive\_slots in 5 minutes: 4.5s

This optimization is currently done by hand but in reality the user will want to just put in the amount of time that they want to get their results in and it should be calculated.

# The equation to determine how to get the probability of a given number of send and receive slots the minimal value of send and receive slots can be found assuming that the send and receive slots need to be equal and the probability of success should be over 98%.

import math

def ProbabilitySuccess(listens: int, sends: int, blocks: int):

prob = listens / blocks

sends = float(sends)

prob = 1 - math.pow(1 - prob, sends)

return prob

def MinNumberOfSendReceiveSlots(

totalTimeForSuccess: int, blockSize: int, probability: float = 0.97

):

# calculate the number of blocks

blocks = int(totalTimeForSuccess // blockSize)

for \_ in range(1, blocks):

# now we run the probability values from 1 to the total number of simulations and create a list of probabilities

prob = ProbabilitySuccess(\_, \_, blocks)

# check if the probability is greater than 97%

if prob > 0.97:

# if it is, return the number of send and receive slots

return \_, \_

return "No solution found"

# total number of slots and simulations

minSendReceive = MinNumberOfSendReceiveSlots(300000, 30, 0.97)

print(f"Minimum number of send and receive slots: {minSendReceive}")

result:

>Minimum number of send and receive slots: (187, 187)

This shows that the optimal send and receive slots are 187 slots at 30 ms each for a 5 minute window. This is a total send\_receive time in 5 minutes of 187\*30ms = 5.6s. of total send and receive time for each device.

Multiple devices:

The mesh consists of many relays to get it to travel the requisite distance for the use case. For example on a farm the signal might have to travel 200 meters total each ble device can send a signal at a distance of around 50 meters with an antenna so a total of 3 devices would be needed to get the signal including the phone receiver. Each relay would be independent of the previous relay and the message can be stored in a device and wait until it is connected to the next device to send the signal. So each send and receive pair can take a total of 5 minutes.

Phone

To run this model the total time would have to be divided by the number of relays:

# total number of slots and simulations

minSendReceive = MinNumberOfSendReceiveSlots(100000, 30, 0.97)

result:

> Minimum number of send and receive slots: (108, 108)

Therefore 108 \* 30ms slots = 3.24s out of 100 seconds with a duty cycle of 3.24%

The function can be updated with the number of relays:

def MinNumberOfSendReceiveSlots(

totalTimeForSuccessMS: int,

numberOfRelays: int,

blockSize: int,

probability: float = 0.97,

):

# calculate the number of blocks

blocks = totalTimeForSuccessMS // blockSize

blocks = blocks // numberOfRelays

for \_ in range(1, blocks):

# now we run the probability values from 1 to the total number of simulations and create a list of probabilities

prob = ProbabilitySuccess(\_, \_, blocks)

# check if the probability is greater than 97%

if prob > 0.97:

# if it is, return the number of send and receive slots

return \_, \_

return "No solution found"

result:

* Minimum number of send and receive slots: (108, 108)

The final step is to implement the send and receive slots. The code will have to take in the total time before success, the number of relay nodes, the block size and the probability.

From there it should use the equation to find the number of slots needed. With that information a random number can be generated without repeats for the number of blocks between zero and the total number of available slots. This can then be sorted. Now the time for sending each block is given. To implement a sleep function can be used. The device can sleep for the interval or the time between each block so that it sends at the time that was randomly calculated.:

def probability\_of\_success(self, listens: int, sends: float, blocks: int):

prob = listens / blocks

prob = 1 - math.pow(1 - prob, sends)

return prob

def min\_number\_of\_slots(self):

blocks = self.\_total\_time\_before\_success\_ms // self.\_block\_size

blocks = blocks // self.\_number\_of\_relays

for \_ in range(1, blocks):

prob = self.probability\_of\_success(listens=\_, sends=\_, blocks=blocks)

if prob >= self.\_probability:

return \_

def find\_delay\_values(self):

new\_table = []

increment = 0

for i in self.\_send\_blocks:

if increment < len(self.\_send\_blocks) - 1:

increment += 1

left = i

right = self.\_send\_blocks[increment]

delay = right - left

new\_table.append(delay)

self.\_send\_blocks = new\_table

return self

# given the number of send blocks find a random value to send the message

# out of 30\_0000 ms blocks that last around 30 ms each we have to pick 135 random numbers

def create\_send\_blocks(self):

slots = self.min\_number\_of\_slots()

if slots != None:

while len(self.\_send\_blocks) < slots:

send\_block = random.randrange(0, self.\_total\_time\_before\_success\_ms)

if send\_block not in self.\_send\_blocks:

self.\_send\_blocks.append(send\_block)

self.\_send\_blocks = sorted(self.\_send\_blocks)

self.find\_delay\_values()

return self

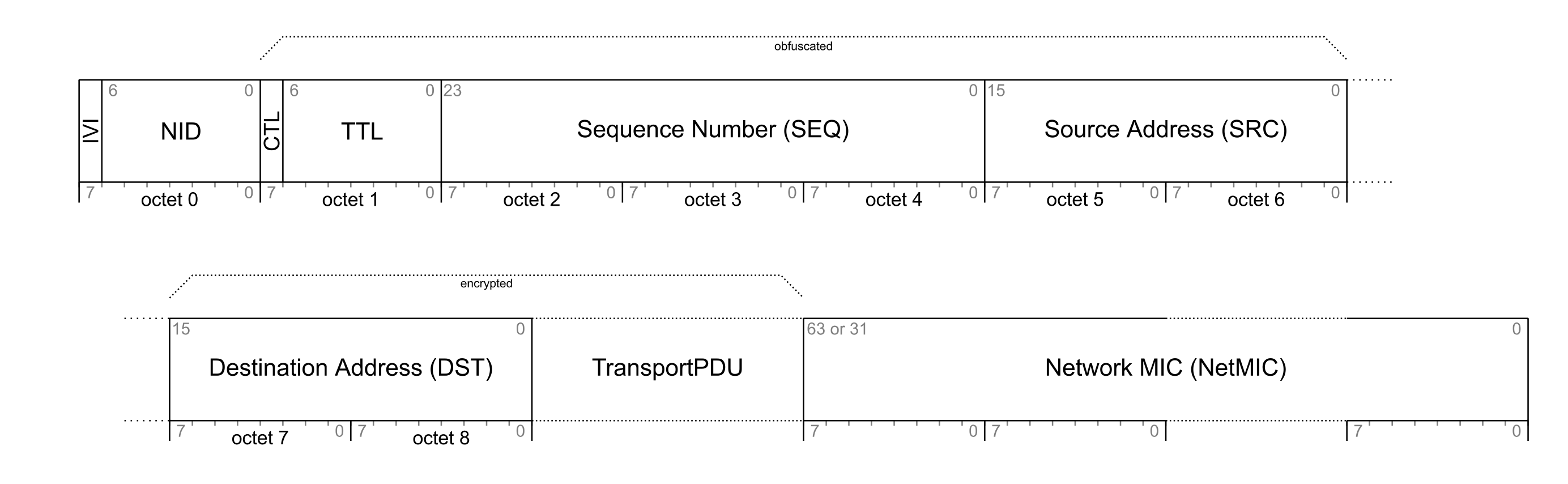
The mesh protocal works on a layers approach much like a standard internet network:

**Graphical user interface, application

Description automatically generated**

The relay protocal works on the network layer

The network message PDUs is:



The network message PDU is both obfuscated and encrypted. The CTL, TTL, SEQ, and SRC is obfuscated that utilizes AES to encrypt the header information. It makes the tracking of nodes difficult.

The destination address and transport PDU is encrypted. The NetMIC is used to authenticate that the DST and Transport PDU is unchanged.

The transport PDU is the meat of the message that the relay does not change.

Obfusication:

A screenshot of a diagram

Description automatically generated

The equation for obfuscate is:

Step 1:

Step2:

Step3:

Where ciphertext = e(key, plaintext) use FernetKey in python

Step 4:

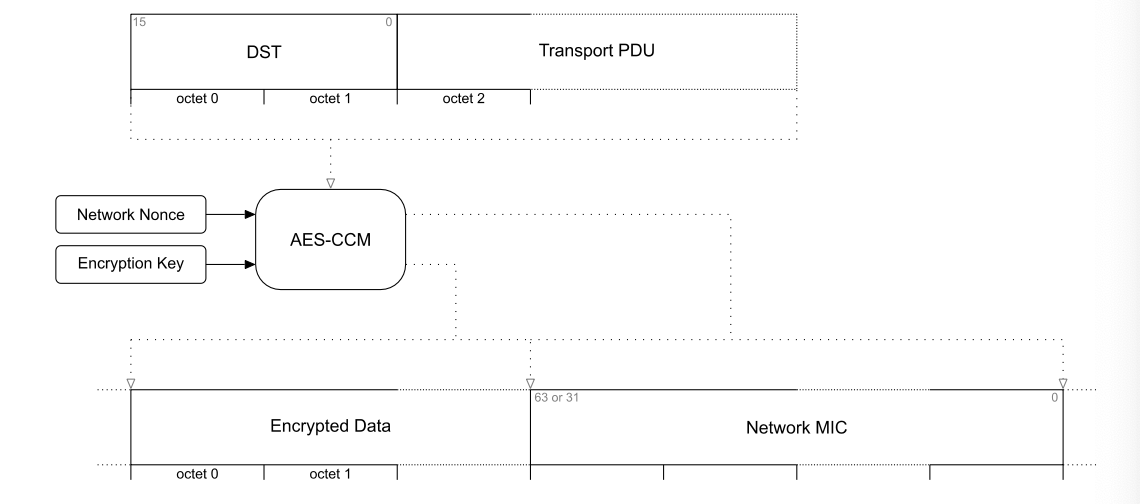
Equation to reverse Obfuscations:

Step 1: same as above

Step 2: same as above

Step 3: same as above

Step 4:



Encrpyt the DST, TransportPDU , and NetMIC using the following equation:

The encryption key has to be derived from the network key!

Code:

PECB is the AES encrypt and decrypt standard using the ECB method. The code takes the first 6 bytes as its solution:

rom Crypto.Cipher import AES

from Crypto.Util.Padding import pad,unpad

def decrypt\_data\_ecb(key, ciphertext):

cipher = AES.new(key, AES.MODE\_ECB)

plaintext = unpad(cipher.decrypt(ciphertext), AES.block\_size)

return plaintext

def encrypt\_data\_ecb(key, plaintext):

cipher = AES.new(key, AES.MODE\_ECB)

ciphertext = cipher.encrypt(pad(plaintext, AES.block\_size))

return ciphertext

# Example usage

key = bytes.fromhex("8b84eedec100067d670971dd2aa700cf") # 16 bytes = 128 bits

plainText = bytes.fromhex("000000000012345678b5e5bfdacbaf6c")

# Encrypt the plaintext

ciphertext = encrypt\_data\_ecb(key, plainText)

print(f"Ciphertext (hex): {ciphertext.hex()}")

plaintext = decrypt\_data\_ecb(key, ciphertext)

print(f"Plaintext (hex): {plaintext.hex()}")

>Ciphertext (hex): 6ca4875075640101a9bf0358d698cb247e1ba4becd78a1741240f6e18e623a43

Plaintext (hex): 000000000012345678b5e5bfdacbaf6c

ExampleData

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| IVI | NID | CTL | TTL | SEQ | SRC |
| 0b0 | 0b1101000 | 0b1 | 0b0000000 | 0x000001 | 0x1201 |

|  |  |  |
| --- | --- | --- |
| DST | TransportPDU | NetMIC |
| oxfffd | ox034b50057e400000010000 | ox035444ce83a670df |

IVI = Least significant bit of the IVI

NID= Derived from the network key in conjunction with the encryption and privacy key.



CTL= 0 if part of the access message(32 bit) 1 if part of the control message (64 bit) NetMIC

TTL = 0 is the message has not been relayed

SEQ = 0x1201 24 bit integer unique for each PDU (random value)

DST = Untouched by relay node. It is the destination fields which is set by originating node and helps to identify that the elements this Network PDU is forwarded to

TransportPDU= the PDU package that is being sent by the network layer

NetMIC = used to authenticate the DST and Transport PDU. This is set by the network layer at each node that transmits or relays this network PDU. It is created using the rest of the packets.

Steps for the BLE Relay Device

A diagram of a network system

Description automatically generated

Logical Flow for Receiving a Network PDU (3.4.6.3)

1. Does the NID value match a known NID? If no ignore
2. Decrypt data
3. De-obstruct the data
4. Does a net key verify the MIC? If no ignore
   1. Use the CCM function (3.8.2.3) to decrypt with own netkey? Having issues with this.
5. Is message in network message cache? If yes ignore
6. Process by lower Transport Layer in our case the Advertisement Bearer on the device

Example for logical flow

Message Sent = 68eca487516765b5e5bfdacbaf6cb7fb6bff871f035444ce83a670df

Message Bytes =

\x68\xec\xa4\x87\x51\x67\x65\xb5\xe5\xbf\xda\xcb\xaf\x6c\xb7\xfb\x6b\xff\x87\x1f\x03\x54\x44\xce\x83\xa6\x70\xdf

|  |  |  |  |
| --- | --- | --- | --- |
| IVI+NID | Obfuscated | Encrypted | NetMic(lasat 4 bytes) |
| 68 | xec\xa4\x87\x51\x67\x65 | xb5\xe5\xbf\xda\xcb\xaf\x6c\xb7\xfb\x6b\xff\x87\x1f\x03\x54\x44\xce | \x83\xa6\x70\xdf |

Steps in reality

1. Convert to bytes and separate into blocks of (IVI+NID, Obfuscated, Encrypted, NETMIC)
2. Does the NID value match a known NID? If no ignore
   1. IVI+NID = 01101000 IVI =0 NID = 1101000
   2. NID in the current Device is IVI=0 NIC =1101000 so it matches
3. Decrypt data
4. De-obstruct the data
   1. De-0bfusicate xec\xa4\x87\x51\x67\x65 with private key
   2. IV Index is known and set throughout the network 12345678
5. Does a net key verify the MIC? If no ignore
   1. Use the CCM function (3.8.2.3) to decrypt with known netkey?
   2. Generate EncKey from NetKey (3.8.6.3.1) 
6. Is message in network message cache? If yes ignore
7. Process by lower Transport Layer in our case the Advertisement Bearer on the device

Advertisement Bearer if relay is authorized:

1. Does the AB have the relay feature?
2. Is the TTL a value of 2 or greater & destination is not unicast of an element of this node.
   1. Decrement TTL
3. Tag Network PDU as a relay
4. Retransmit
   1. Use a small random delay to avoid collisions

Logical Flow for Transmitting a Network PDU:

Step1: Set IVI Field to least significant bit of the IV Index

Step 2: The NID field set to the proper value with the devices Enryption key and Privacy Key

Step 3: The CTL Field is kept the same

Step4: The TTL field is kept the same

Step5: The SEQ field should be incremented by 1

Step6: The SRC field is set to the current Unicast address of the current node

Step7: The DST is the unicast, group or virtual address which is set by another layer so it is. Kept the same

Step8: The transport PDU is untouched

Step9: The NetMIC is created

Step10: The message is sent to all network interfaces so it is broadcasted out. It will be sent if it passes the output filter which just checks if the TTL is set to 1

**Mesh Protocol Notes** [**Mesh Protocol | Bluetooth® Technology Website**](https://www.bluetooth.com/specifications/specs/mesh-protocol/)

**Graphical user interface, application

Description automatically generated**

**Uses BLE Core**

**Mesh transports data from one end to another**

**Bearer layer-**

**Network layer**

**Lower**

**Upper**

**Access**

**Foundation model**

**Model**

**Provisioning-**

**Provision Bearer**

**Provision Transport**

**Provision Protocol**

**Keys:**

**APPKey- only used for that application (light v security)**

**NetKey- Secures on the network layer. All in a subnet share the same netkey. Use this to derive the NetworkEncryptionKey and Privacy Key.**

**DevKey- Each device has a unique DebKey by device and provisioner.**

**Diagram

Description automatically generated**

**Provisioner Process-**

PB-ADV is an advertisement protocol. The provisionee sends a message out in BLE saying it wants to be provisioned.

Provisioner uses the client role and the provisionee is the server role

Table

Description automatically generated

Text

Description automatically generated with low confidence

AD-Type- 0x29

Transaction number start at 0 increment by 1

Link ID is used to identify a link between the devices

Generic PDU: (pg 577 and example on pg 706)

1. Provisioning Invite
2. Provisioning Capabilities
3. Provisioning State
4. Provisioning Public Key
5. Provisioning Input Complete
6. Provisioning Confirmation
7. Provisioning Random
8. Provisioning Data
9. Provisioning Complete
10. Provisioning Failed

Diagram

Description automatically generated

PB-ADV Provisioning Invite

The provisioner invites the new device to join the network. This includes the duration of time the new device will attract(beep, flash, etc)the device to be provisioned.

Provisioning Invite

Attention Duration : 00 (0 seconds)

Message : 0000

LinkID : 23af5850

Transaction : 00

Segment0 : 0000

SegN : 00

TotalLength : 0002

FCS : 14

Message0 : 23af585000000002140000

\*Provisioing PDU How it works is you take the PB-ADV PDU Field as follows:

PB-ADV-PDU Field(Octets):

|  |  |  |
| --- | --- | --- |
| LinkID(4) | TransactionNumber(1) | GenericPDU(1-24) |

GenericPDU: in the priviosioning uses the Generic Provisioning PDU:

|  |  |
| --- | --- |
| GenericProvisioningControl | GenericProvisiongPayload |

GenericProvisioningControl:

Use Section 5.3.1 (TX start PDU, TX Ack PDU, TX ContinuationPDU, ProvBearerControl PDU, LinkOpenMessage…)

GenericProvisioningPayload also called Provisioning PDU:

Use Section 5.4.1(ProvisioningPDU, Provisioning Invite, prov capabilities, provstart, provkeys…)

|  |  |  |
| --- | --- | --- |
| **LinkID(32)** | **TransactionNumber** | GenericPDU |
| 23af5850 | **000** |  |
| **00100011101011110101100001010000** | **00000000** |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| SegN(6) | GPCF(2) | TotalLength (16) | FCS(8) | Padding(2) | Type(6) | Parameters(1) |
| 0x00 | 00 | 0002 | 14 | 0b00 | 0x00 | 0x00 |
| 000000 | 00 | 0000000000000010 | 00010100 | 00 | 000000 | 00000000 |
|  |  |  |  |  |  |  |

**Generic Provisioning Layer**

The generic provisioning layer is responsible for transport of Generic Provisioning PDUs over an

unreliable connectionless provisioning bearer. This layer also defines Generic Provisioning PDUs.

Graphical user interface, text, application, email

Description automatically generated

**Generic Provisioning Control**

**Transaction Start PDU**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SegN | GPCF | TotalLength | FCS | DATA |
| 00 | 00 | 0002 | 14 | ProvsioningPDU |
|  |  |  |  |  |

Table

Description automatically generated

When transmitted using PB-ADV, the FCS field is calculated as defined by 3GPP TS 27.010 with the

Polynomial (x8 + x2 + x1 + 1) and is calculated over the Provisioning PDU only.

A provisioner sends a provisioning invite PDU to indicate to the provisionee that the provisioning has started.

Provisioning Payloads:

Provisioning PDU: Communicate between a provisioner and provisionee:

Table

Description automatically generated

Table

Description automatically generated

Parameters- See below (Provisioning invite, etc)

Provisioning Invite: Attention Duration (Attention Timer State- 1 octet: 00) pg 577

* 00 = off
* 0x01-0xFF remaining time in seconds

**Provisioing Acknowledge**

Now that the provisioner has sent an invite to the provisionee and the provisioning is started an acknowledge message is sent to acknowledge the invite transation. We always have to acknowledge a provisioning PDU

PBADV Transaction Acknowledge

LinkID : 23af5850

Transaction : 00

Message : 23af58500001

GenericProvisioningControlField:

|  |  |  |
| --- | --- | --- |
| **LinkID(32)** | **TransactionNumber** | GenericPDU |
| 23af5850 | **000** |  |
| **00100011101011110101100001010000** | **00000000** |  |

GenericPDU(Transaction Acknowledgment PDU)

Graphical user interface, text, application, email

Description automatically generated

|  |  |
| --- | --- |
| Padding (6)bits | GPCF(2)bits |
| 0b000000 | 01 |

The next step is for the provisionee to send its capabilities to the provisioner

**PB-ADV Provisioning Capabilities**

The new device responds to the invite by sending the following:

Number of Elements : 01

Algorithms : 0001

Public Key Type : 00

OOB Type : 00

Output OOB Size : 00

Output OOB Action : 0000

Input OOB Size : 00

Input OOB Action : 0000

Message : 010100010000000000000000

LinkID : 23af5850

Transaction : 80

Segment0 : 010100010000000000000000

SegN : 00

TotalLength : 000c

FCS : d6

Message0 : 23af58508000000cd6010100010000000000000000

Generic Provision Payload (message)

Provisioning Capabilities

Table

Description automatically generated

4- CapabilitiesParameter

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| NumberOf Elements (8 bits) | Algorithms(16 bits) | PKType | OOBType | OOBSize | OOBAction | InputOOBSize | inputOOBAction |
| 0x01 | 0x0001 | 0x00 | 0x00 | 0x00 | 0x0000 | 0x00 | 0x0000 |
| 00000001 | 0000000000000001 | 00000000 | 00000000 | 00000000 | 0000000000000000 | 00000000 | 0000000000000000 |

3-Provisioning PDU

|  |  |  |
| --- | --- | --- |
| Padding | Type | Parameters |
| Ob00 | 0x01 | See above |
| 00 | 000001 | See above |

2-GenericProvisioningcontrolField (GenericPDU)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SegN(bytes | GPCF(bytes) | TotalLength | FCS | DATA |
| 00 | 00 | 00c | d6 | ProvsioningPDU |
| 000000 | 00 | 000000001100 | 11010110 |  |

1-Transaction Start

|  |  |  |
| --- | --- | --- |
| **LinkID(32)** | **TransactionNumber** | GenericPDU |
| 23af5850 | **80** |  |
| **00100011101011110101100001010000** | **10000000** | **above** |

**Now we acknowledge the PB-ADV Transaction that just took place**

**Provisioner->Provisionee (The provisioner ackn that they got the message) The provisioner shall calculate the FCS for the PDU and if it matches the FCS in the startPDU it shall send this transacation ack PDU**

**FCS is calculated over the Provisioning PDU only**

Message = 010100010000000000000000

FCS(provisioning PDU only) = x^8 + x^2 +x^1 +1

Table

Description automatically generated

PBADV Transaction Acknowledge since when a sender sends a message the other device has to acknowledge it.

When the first message in a transaction is sent, the sender shall start the acknowledgment timer with an initial value of 30 seconds. When the Transaction Acknowledgment message for the transaction is received, the sender shall stop the acknowledgment timer. When the acknowledgment timer expires, the sender shall cancel the transaction, cancel the provisioning process, and close the link.

LinkID : 23af5850

Transaction : 80

Message : 23af58508001

1-Transaction Start

|  |  |  |
| --- | --- | --- |
| **LinkID(32)** | **TransactionNumber** | GenericPDU |
| 23af5850 | **80** |  |
| **00100011101011110101100001010000** | **10000000** | **above** |

|  |  |
| --- | --- |
| Padding (6)bits | GPCF(2)bits |
| 0b000000 | 01 |

**PB-ADV-Provisioning Start:**

After the provisioner receives the capabilities send a provisioning start message stating the algorithm and authentication method used but no keys or anything

Provisioning Start

Algorithm : 00

Pub Key Type : 0000

Authentication Method : 00

Authentication Action : 00

Authentication Size : 00

Message : 020000000000

LinkID : 23af5850

Transaction : 01

Segment0 : 020000000000

SegN : 00

PDU Length : 0006

FCS : 64

Message0 : 23af58500100000664020000000000

PB-ADV

|  |  |  |
| --- | --- | --- |
| **LinkID(32)** | **TransactionNumber** | GenericPDU |
| 20x3af5850 | **0x01** | **GenericProvPDU** |
|  |  | **TransactionStartPDU** |

TransactionStartPDU

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SegN(bytes | GPCF(bytes) | TotalLength | FCS | DATA |
| 0x00 | 0x00 | 0x0006 | 0x64 | ProvsioningPDU |
| 000000 | 00 | 000000001100 | 11010110 |  |

Provisioning PDU

|  |  |  |
| --- | --- | --- |
| Padding | Type | Parameters |
| 0b00 | 0x02 |  |

Provisioning Start Parameters(see table 5.4.1.3):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Algorithm** | **PublicKey** | **AuthMethod** | **AuthAction** | **AuthSize** |  |
| 0x00 | **0x00** | **0x00** | **0x00** | **0x00** |  |

**PB-ADV- Transaction-ack**

**Acknowledge but note that the transaction is 01 that is being ack**

PBADV Transaction Acknowledge

LinkID : 23af5850

Transaction : 01

Message : 23af58500101

**PB-ADV Provisioning Public Key (Provisioner)**

The Provisioner sends its public key to the new device. This message contains the 512-bit public key, and is therefore transmitted using three segments in three messages.

Provisioning Start

Public Key : 2c31a47b5779809ef44cb5eaaf5c3e43d5f8faad4a8794cb987e9b03745c78dd919512183898dfbecd52e2408e43871fd021109117bd3ed4eaf8437743715d4f

Message : 032c31a47b5779809ef44cb5eaaf5c3e43d5f8faad4a8794cb987e9b03745c78dd919512183898dfbecd52e2408e43871fd021109117bd3ed4eaf8437743715d4f

LinkID : 23af5850

Transaction : 02

Segment0 : 032c31a47b5779809ef44cb5eaaf5c3e43d5f8fa

Segment1 : ad4a8794cb987e9b03745c78dd919512183898dfbecd52

Segment2 : e2408e43871fd021109117bd3ed4eaf8437743715d4f

SegN : 02

PDU Length : 0041

FCS : d1

Message0 : 23af585002080041d1032c31a47b5779809ef44cb5eaaf5c3e43d5f8fa

Message1 : 23af58500206ad4a8794cb987e9b03745c78dd919512183898dfbecd52

(06=00000110) segment index=1 (segment#1)

Message2 : 23af5850020ae2408e43871fd021109117bd3ed4eaf8437743715d4f

0a=2 for segment 2

Note this is three messages sent in three segments just because it is large. How this works is that the transaction is incremented to 02 the SegN is 02 since we have three messages the length is 0041= 65 integer for the length of the messages. The message transaction start is started then the continuation messages sets index to 1 then 2 etc…

Provisioning PDU

|  |  |  |
| --- | --- | --- |
| Padding | Type | Parameters |
| 0b00 | 0x03 |  |

Parameters

Public keyX then y

|  |
| --- |
| 2c31a47b5779809ef44cb5eaaf5c3e43d5f8faad4a8794cb987e9b03745c78dd919512183898dfbecd52e2408e43871fd021109117bd3ed4eaf8437743715d4f |

We have to segment this(65 octets total divide by 3 makes

|  |
| --- |
| 032c31a47b5779809ef44cb5eaaf5c3e43d5f8faad4a8794cb987e9b03745c78dd919512183898dfbecd52e2408e43871fd021109117bd3ed4eaf8437743715d4f |

20 23 22

Now

Message1

PB-ADV

|  |  |  |
| --- | --- | --- |
| **LinkID(32)** | **TransactionNumber** | GenericPDU |
| 20x3af5850 | **0x02** | **GenericProvPDU** |
|  |  | **TransactionStartPDU** |

TransactionStartPDU

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SegN(bytes | GPCF(bytes) | TotalLength | FCS | DATA |
| 0x02 | 0x00 | 0x0041 | 0xd1 | ProvsioningPDU(message1) |

Message2

PB-ADV

|  |  |  |
| --- | --- | --- |
| **LinkID(32)** | **TransactionNumber** | GenericPDU |
| 20x3af5850 | **0x02** | **GenericProvPDU** |
|  |  | **TransactionContinuationPDU** |

Transaction Continuation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SegmentIndex(4 bits) | Buffer | Data |  |  |
| 0001 | 0b10 | ProvsioningPDU(message2) |  |  |

Message2

PB-ADV

|  |  |  |
| --- | --- | --- |
| **LinkID(32)** | **TransactionNumber** | GenericPDU |
| 20x3af5850 | **0x02** | **GenericProvPDU** |
|  |  | **TransactionContinuationPDU** |

Transaction Continuation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SegmentIndex(4 bits) | Buffer | Data |  |  |
| 000010 | 0b10 | ProvsioningPDU(message3) |  |  |

**PB-ADV Provisioning Public Key (Device)**

The device sends its public key to the Provisioner. This is again a multi-segment transaction.

Provisioning Start

Public Key : f465e43ff23d3f1b9dc7dfc04da8758184dbc966204796eccf0d6cf5e16500cc

0201d048bcbbd899eeefc424164e33c201c2b010ca6b4d43a8a155cad8ecb279

Bluetooth SIG Proprietary Page **324 of** 333Mesh Profile / Specification

Message : 03f465e43ff23d3f1b9dc7dfc04da8758184dbc966204796eccf0d6cf5e16500

cc0201d048bcbbd899eeefc424164e33c201c2b010ca6b4d43a8a155cad8ecb2

79

LinkID : 23af5850

Transaction : 81

Segment0 : 03f465e43ff23d3f1b9dc7dfc04da8758184dbc9

Segment1 : 66204796eccf0d6cf5e16500cc0201d048bcbbd899eeef

Segment2 : c424164e33c201c2b010ca6b4d43a8a155cad8ecb279

SegN : 02

PDU Length : 0041

FCS : 10

Message0 : 23af5850810800411003f465e43ff23d3f1b9dc7dfc04da8758184dbc9

Message1 : 23af5850810666204796eccf0d6cf5e16500cc0201d048bcbbd899eeef

Message2 : 23af5850810ac424164e33c201c2b010ca6b4d43a8a155cad8ecb279