**Laura Protocol/Platform Specification**

**Version 0.1**

**Created By: Phil Herlihy**

**License:**

**The MIT License (MIT)**

**Copyright (c) 2013-2016 Philip Herlihy**

**Permission is hereby granted, free of charge, to any person obtaining a copy**

**of this software and associated documentation files (the "Software"), to deal**

**in the Software without restriction, including without limitation the rights**

**to use, copy, modify, merge, publish, distribute, sublicense, and/or sell**

**copies of the Software, and to permit persons to whom the Software is**

**furnished to do so, subject to the following conditions:**

**The above copyright notice and this permission notice shall be included in**

**all copies or substantial portions of the Software.**

**THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.**

**Network Modes:**

***Star* (Trusted Mode, Default)** – Trusted Node UUIDs are on a whitelist on the Gateway. No untrusted Nodes and no open join. Communications are encrypted. All communications flow through the Gateway.

**Star(Untrusted Mode)** – Nodes can join and leave freely. Communications are encrypted, but anyone with a Laura Module can join the network to receive and send messages. All communications flow through the Gateway.

**Point-To-Point (Trusted Mode) –** Two Nodes can locally pair with one another for Point to Point communication. No Gateway necessary.Communications are encrypted, but the keys must be exchanged via USB (no exchange via radio link).

***(Preliminary) Star-Mesh* (Untrusted Mode)** - Nodes can hop messages across other Nodes to reach the Gateway. The network’s overall distance is improved based around how many Nodes there are in a given area. Nodes can join and leave freely. Communications are encrypted, but network type also vulnerable to Man-In-The-Middle style attacks, given the trust of other Nodes.

***(Preliminary) Star-Mesh* (Trusted Mode)** - Nodes can hop messages across other Nodes to reach the Gateway. The network’s overall distance is improved based around how many Nodes there are in a given area. Trusted Node UUIDs are on a whitelist on the Gateway. No untrusted Nodes and no open join. Communications are encrypted, but network type also vulnerable to Man-In-The-Middle style attacks, given the trust of other Nodes. Slightly better than Untrusted, as Nodes are ones that are explicitly granted access to the Network.

**Network Features:**

Open Join (Star)/Closed Join (Star)

Peer to Peer Key Exchange

Node Management

Individual Node Keys (Node communication isolation)

**Gateway (Class):**

This is the central point of the Star-only type network. It sits at the center and acts as the central point of communication for all of the nodes. Encrypted transactions flow through and are facilitated by the Gateway. Any buffered or queued messages are stored in a section of it's encrypted storage. The message encrypted storage key is randomly generated on startup, so messages waiting in the queue are not preserved through reboots. The Keystore is also stored in a separate encrypted partition, which is can be configured to be either ephemeral or non-ephemeral. In the case of the ephemeral setup, node re-pairing is required on every startup of the Gateway. In the case of the non-ephemeral setup, the partition is locked with the administrator's password. Their presence is required on startup to unlock the partition with the Keystore, in this configuration. In Star-only mode, the Gateway is the authority of the network. The Nodes authenticate, and remain synchronized with the Gateway.

The Key Exchange Public/Private keypair are generated by the software when there is no Keystore detected in the encrypted partition. In non-ephemeral mode, this keypair will remain in the encrypted partition until it is destroyed.

The Gateway software creates an operational database during it's service. This database is stored in the encrypted partition with the Keystore. The following tables are the tables used by the Gateway software to track the state of the Nodes, their message queues, and their keys.

**Node Info Table:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Node UUID | Node Network ID | Last Sync Secret | Next Sync Secret | Last Message Timestamp (UTC) | Node Class |
| Permission To Broadcast | Last Message RSSI | Broadcasts Submitted | Node State | Registration Date | Expiry Date |

**Untouchables Table:**

|  |  |  |
| --- | --- | --- |
| Node ID | Untouchable Expiry | Date marked Untouchable |

**Node Outgoing Message Table:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Node Network ID | Node UUID | Message Timestamp | Transmission Attempts | Message |

**Node Incoming Message Table:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Destination Node ID | Source Node ID | Message Sequence | Total MessageSequence | Transmission Attempts | Message Timestamp | Message |

**General Statistics Table:**

|  |  |  |
| --- | --- | --- |
| Messages In Last 24Hrs | Node IDs Registered | Time Since Last Message In |

**Gateway Configuration Table:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Max Node ID | Network ID | Allow Broadcasts | Gateway UUID | Trusted Mode | Gateway State | Broadcast Limit |

**Trusted Table:**

|  |  |  |
| --- | --- | --- |
| Node UUID | Trusted | Preferred Node ID |

Incoming messages from the Node are decrypted with the Node Secret Key. Outgoing messages to the Node are encrypted with the Node Secret Key.

Node Secret Keys are generated by the Atmel SAMD21 (leveraging some features of the Atmel ATAES132A), during the Key Generation Phase and exported to the Gateway during the Key Exchange Phase using public/private type key exchange.

The Gateway projects some of it's interface onto it's on board OLED screen. Options for local pairing can be selected using the 5-way pad. It also optionally displays active statistics from the network when idle.

**Node (Class)**:

This class is meant to operate independently, is able to send messages and receive messages entirely on it's own. The Handheld is capable of notifying the user (vibration and buzzer), if placed in a pocket. It stores a single public key for Gateway Key Exchange at a time. In Star mode, the Node will only communicate to the Gateway. In Mesh-only mode, the Node will communicate with other Nodes in untrusted mode.

The Node leverages it's on board ATAES132A for it's encryption features and UUID. The HopeRF modules have built-in support for AES128 encryption, but this is not used on Laura devices. It generates it's Secret Key when it's network mode is selected, and then waits to receive a public key from a Gateway. The Node must regenerate it's Secret Key *every time* it's network mode changes. Once the public key has been installed to the device (stored in the EEPROM on the ATAES132A), it encrypts it's Secret Key with the current Gateway Public Key, and then transmits it to the Gateway. This can happen over radio (via the Gateway Discover feature), or over it's local USB connection.

Once the Secret Key has been sent to the Gateway, and the Gateway sends a KEY-ACK message. From this point on, all of the Node's outgoing messages are encrypted with this Secret Key and sent to the Gateway. All of it's incoming messages are checked for validity, and then decrypted with this Secret Key.

The Gateway also has forceful control of the Node Secret Key. If a device is lost or stolen, the Gateway can send a KEY-DESTROY command. This command has to be authenticated using the current Secret Key (so as to defeat rouge KEY-DESTROY commands), but once the message is authenticated it will destroy the Secret Key, Gateway Public Key and unpair itself from the Gateway.

No messages are queued on the Node. If the Node fails to transmit it's message (or runs out of attempts with no MSG-ACK from the Gateway), it will destroy the message.

**Packet Format:**

Each packet has a static header, and then it’s payload varies according to it’s packet type. The control information and addresses are inside the encrypted area of the payload to prevent tampering and snooping. All control messages have discreet packet sequences to reference their response, i.e. ACKs explicitly acknowledge a specific message sequence number. The Gateway will always sit at Node ID 0.

**Packet Header:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Packet Destination Node ID** | **Source Node ID** | **Packet Type** | **Packet Sequence** | **Sync Secret** |
| 1 Byte | 1 Byte | 1 Byte | 1 Byte | 4 Bytes |

**FIND-GATE:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No Payload** | **No Payload** | **No Payload** | **No Payload** | **No Payload** | **No Payload** |

FIND-GATE is a unique packet, because it’s a broadcast packet, intended to garner a reply from a Gateway listening on the network the broadcast is taking place on.

**Encrypted:** No

**Type:** 0x00

**Expect ACK:** No

**GATE-DESC:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Gateway FQN** | **Gateway Network Mode** | **Gateway Trust Setting** | **Gateway Join Setting** | Reserved |
| 16 Bytes | (4 Bits) | (1 Bit) | (2 Bit) | (1 Bit) |

GATE-DESC contains the important information about an operating Gateway. The Node also collects the Network ID while it’s running it’s FIND-GATE queries, so this does not need to be supplied in the GATE-DESC.

**Encrypted:** No

**Type:** 0x01

**Expect ACK:** No

**JOIN-GATE**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No Payload** | **No Payload** | **No Payload** | **No Payload** | **No Payload** | **No Payload** |

JOIN-GATE is simply the request for the Node to join the Gateway. It can receive a NACK (if the Gateway is closed), WAIT (if the Gateway requires approval) or ACK (which will move along the key-exchange process).

**Encrypted:** No

**Type:** 0x02

**Expect ACK:** Yes (Upon success)

**GATE-PUB-KEY:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TBD** |  |  |  |  |
| **TBD** |  |  |  |  |

GATE-PUB-KEY packets contain the Gateway Public Key that will be used to encrypt the Node Secret Key, for transmission to the Gateway.

**Encrypted:** No

**Type:** 0x03

**Reply with ACK:** No

**NODE-SECRET:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Requested Node ID** | **Node Secret Key** | **Node UUID** |  |  |
| **1 Byte** | **32 Bytes** | **8 Bytes** |  |  |

NODE-SECRET packets contain the Node Secret Key, it’s UUID and it’s Requested ID, if it has one. If it does not, this field is 0, indicating it will be happy with whatever the Gateway gives it. The Gateway can deny the requested ID, if it is unavailable.

**Encrypted:** Yes (Gateway Public Key)

**Type:** 0x04

**Expect ACK:** Yes (KEY-ACK)

**KEY-ACK:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Actual Node ID** | **Reserved** | **Reserved** | **Reserved** |  |
| 1 Byte | 1 Byte | 1 Byte | 1 Byte |  |

A KEY-ACK is transmitted in response to the Gateway receiving and successfully storing a Node’s unique data and registering it’s Node ID. A KEY-ACK expects the Node to ACK it, indicating that the Network Join transaction is complete.

**Encrypted:** Yes (Node Secret Key)

**Type:** 0x05

**Expect ACK:** Yes

**KEY-DESTROY:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Time To Destroy** | **Reserved** |  |  |  |
| 4 Bytes | 1 Byte |  |  |  |

A KEY-DESTROY packet is sent to a Node when it’s to be kicked from the Network. Receiving this directive will cause the Node to destroy it’s current Node Secret Key and exit the network, relinquishing it’s Node ID. Time to Destroy is a timer that will tick down before destroying the key. In this case, the Node will reply with an ACK, and then the Gateway will wait for the NODE-EXIT when the timer expires.

**Encrypted:** Yes (Node Secret Key)

**Type:** 0x06

**Expect ACK:** Yes (If Time to Destroy is greater than 0)

**NODE-EXIT:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Relinquished Node ID** | **Reserved** |  |  |  |
| 1 Byte | 1 Byte |  |  |  |

A NODE-EXIT packet is sent from a Node to the Gateway when it’s key has been destroyed, and it is relinquishing control of it’s Node ID. This can happen on a voluntary exit, or an involuntary exit (KEY-DESTROY).

**Encrypted:** Yes (Node Secret Key)

**Type:** 0x07

**Expect ACK:** Yes

**PAIR-REQ:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Node Secret Key** | **Node UUID** | **Sync Secret Seed** |  |  |
| 32 Bytes | 8 Bytes | 4 Bytes |  |  |

A PAIR-REQ message is only ever acknowledged via USB link. This message type is not used over the radio link for security reasons. This packet is used to facilitate P2P local key exchange, it is not used in any other network mode.

**Encrypted:** No

**Type:** 0x08

**Expect ACK:** Yes (Upon success)

**ACK:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Acknowledged Packet Sequence** | **Reserved** |  |  |  |
| 1 Byte | 1 Byte |  |  |  |

This packet is used as an acknowledgement that a packet was received and accepted. It is especially important for bulk transfers, or any blocks which require multiple packets and synchronization.

**Encrypted:** Yes (Node Secret Key)

**Type:** 0x09

**Expect ACK:** No

**NACK:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Acknowledged Packet Sequence** | **Reserved** |  |  |  |
| 1 Byte | 1 Byte |  |  |  |

This packet is used as an non-acknowledgement that a packet was received and accepted. It is especially important for bulk transfers, or any blocks which require multiple packets and synchronization.

**Encrypted:** Yes (Node Secret Key)

**Type:** 0x01

**Expect ACK:** No

**NODE-MSG:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Message Destination Node ID** | **Message Sequence** | **Total Sequence** | **Payload Length** | **Reserved** | **Message Payload** |
| 1 Byte | 1 Byte | 1 Byte | 1 Byte | 1 Byte | n Bytes (< BYTES\_MAX) |

A NODE-MSG packet is used to transfer a message. The destination listed here is not a duplicate from the header, the header’s destination is the Gateway (Node 0), and this destination is the Node ID where the message is intended to arrive.

**Encrypted:** Yes (Node Secret Key)

**Type:** 0x0A

**Expect ACK:** Yes (NACK on NO ADDRESS)

**NODE-BROADCAST:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Message Sequence** | **Payload Length** | **Message Payload** |  |
| 1 Byte | 1 Byte | n Bytes (< BYTES\_MAX) |  |

A NODE-BROADCAST packet is used to transfer a message that will be forwarded to all of the Nodes. The Gateway will check if broadcast packets for are allowed for all Nodes, or Nodes on the broadcast whitelist.

**Encrypted:** Yes (Node Secret Key)

**Type:** 0x0B

**Expect ACK:** Yes (NACK on NO PERMISSION)

**GATE-RANGE-CHK:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No Payload** | **No Payload** | **No Payload** | **No Payload** | **No Payload** | **No Payload** |

The GATE-RANGE-CHK is used as a estimated range/signal sentinel. It is sent to the Gateway (the Gateway will record the Node’s RSSI) and then reply with an ACK, which will allow the Node to determine it’s RSSI. It will also be an obvious indicator of being in range of the Gateway, the Node will not receive an ACK if it’s not in range (or has left range since transmitting).

**Encrypted:** Yes (Node Secret Key)

**Type:** 0x0C

**Expect ACK:** Yes

**TIME-SYNC-REQ:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No Payload** | **No Payload** | **No Payload** | **No Payload** | **No Payload** | **No Payload** |

This packet is dispatched from the Client when it needs to resynchronize it’s on board time and sync secret with the Gateway.

**Encrypted:** Yes (Node Secret Key)

**Type:** 0x0D

**Expect ACK:** No (Expects TIME-SYNC)

**TIME-SYNC:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Current UTC Time** | **Sync Secret Seed** | **Reserved** |  |  |
| 8 Bytes | 4 Bytes | 2 Bytes |  |  |

The TIME-SYNC packet is dispatched to the client in reply to a TIME-SYNC-REQ, it contains the current UTC time (accounted for protocol drift), and a freshly generated Sync Secret Seed. This will set up the Node for message exchange.

**Encrypted:** Yes (Node Secret Key)

**Type:** 0x0E

**Expect ACK:** Yes

**TIME-FORCE-SYNC:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Current UTC Time** | **Sync Secret Seed** | **Reserved** |  |  |
| 8 Bytes | 4 Bytes | 2 Bytes |  |  |

 If the Gateway detects an issue or de-synchronization with the Network, it will issue a TIME-FORCE-SYNC packet as a broadcast to all nodes. It contains the same information that a normal TIME-SYNC packet, but is only issued during

**Encrypted:** Yes (Node Secret Key)

**Type:** 0x0F

**Expect ACK:** Yes

**Trusted Command Structure (Star, Trusted):**

**Gateway Discovery and Gateway Key Exchange Over Radio (Gateway --- Node):**

1. (Unauthenticated) <\*--- FIND-GATE
2. (Unauthenticated) ---> GATE-DESC
3. (Unauthenticated) <--- JOIN-GATE
4. (Unauthenticated) ---> ACK
5. (Unauthenticated) ---> GATE-PUB-KEY
6. (Commit Gateway Public Key)
7. (Encrypted, Gateway Public Key) <--- NODE-SECRET
8. (Commit Node Secret Key, Registers Node ID)
9. (*Authenticated, Encrypted*) ---> KEY-ACK
10. *(Authenticated, Encrypted)* <--- ACK
11. (Node is Entered)

**Gateway Range Check (Gateway --- Node):**

1. (Authenticated, Encrypted) <--- GATE-RANGE-CHK
2. (Authenticated, Encrypted) ---> ACK

**Synchronize Time Request (Gateway --- Node):**

1. (Authenticated, Encrypted) <--- TIME-SYNC-REQ
2. (Authenticated, Encrypted) ---> TIME-SYNC
3. (Authenticated, Encrypted) <--- ACK

**Remote Key Wipe (Gateway --- Node):**

1. (Authenticated, Encrypted) ---> KEY-DESTROY
2. (Authenticated, Encrypted) <--- NODE-EXIT
3. (Authenticated, Encrypted) ---> ACK
4. (Node Secret Key destroyed, Gateway Public Key destroyed, Node is Exited)

**Client to Client (Gateway --- Node):**

1. (Authenticated, Encrypted) <--- NODE-MSG
2. (Authenticated, Encrypted) ---> ACK
3. *To Destination Node:*
4. (Authenticated, Encrypted) ---> NODE-MSG
5. (Authenticated, Encrypted) <--- ACK

**Client To Broadcast (Gateway --- Node);**

1. (Authenticated, Encrypted) <--- NODE-BROADCAST
2. (Authenticated, Encrypted) ---> ACK
3. (Gateway verifies permission to Broadcast, Checks maximum Node Broadcast counter, Places Broadcast message in all queues)

**Broadcast Time Synchronization (Gateway --- Node):**

1. (Authenticated, Encrypted) ---> TIME-FORCE-SYNC
2. (Authenticated, Encrypted) <--- ACK

**Node Exit (Gateway --- Node):**

1. (Authenticated, Encrypted) <--- NODE-EXIT
2. (Authenticated, Encrypted) ---> ACK
3. (Node Secret Key destroyed, Gateway Public Key destroyed, Node is Exited)
4. (Gateway places Node ID in the Untouchables pool for set period of time)

**Untrusted Command Structure (Star, Trusted):**

**Gateway Discovery (Gateway --- Node):**

1. (Unauthenticated) <\*--- FIND-GATE
2. (Unauthenticated) ---> GATE-DESC