**LoRaWAN and 4 attacks toward LoRaWAN**

**The setup of a LoRa network**

This experiment uses LoRa® technology evaluation kit - 800 (part number: DV164140-1) from Microchip to set up LoRa networks. The kit is developed to test LoRa transmission and LoRa network performance. The kit includes a LoRa gateway, 2 LoRa motes and an example LoRa server.

The [Microchip LoRa gateway](http://ww1.microchip.com/downloads/en/DeviceDoc/40001827A.pdf) has the LoRa module SX1301, which is the base band processor and data concentrator. The gateway provides communication with the Microchip supported example LoRa network and application server. Uplink messages are issued according to the LoRaWAN specification, and they are captured and forwarded by Microchip’s Gateway. The gateway has 6 channels, and it includes an LCD screen, SD Card for Configuration Data, Ethernet connection, 868 MHz antenna, and full-band capture radios.

The LoRa evaluation kit also includes two RN2483 Mote boards (part number: DM164138). The LoRa motes are LoRa end devices. The LoRa Mote is a demonstration board, and it includes a transceiver module RN2483, which is developed according to LoRa technology. The module accepts commands via UART interface. Communication with the module is achieved through two methods of power supply, USB and Battery.

This experiment uses LoRa servers from [brocaar](https://github.com/brocaar/loraserver). The servers include a LoRa gateway bridge, a LoRa server and a LoRa app server. Also, the backend of [The Things Network](https://www.thethingsnetwork.org/) can also be used to provide server and gateway service. The end device can connect to the TTN’s gateway and server, and data can be observed in the website console of TTN’s application server

In addition, minicom is used to control the serial port communication.

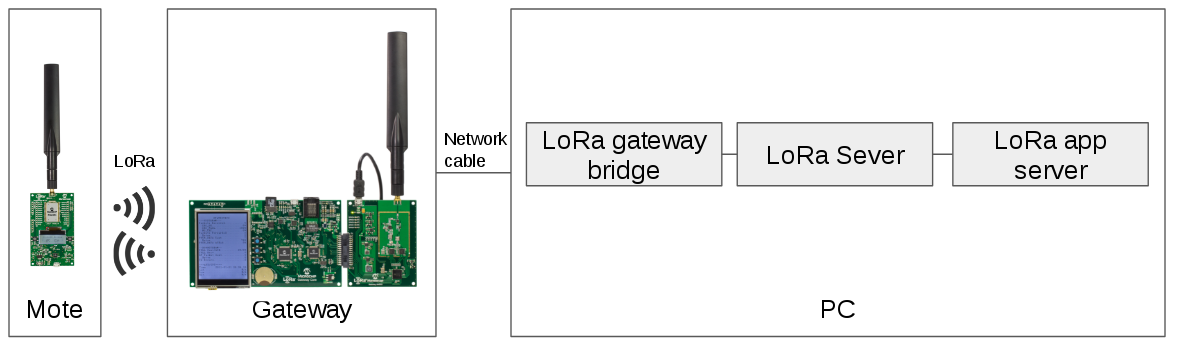


Fig 1: LoRaWAN architecture

The LoRaWAN in this experiment consists of an end device (also called mote), a gateway and LoRa servers.

This is the procedure of the setup:

1. Build the LoRa gateway bridge ([click](https://github.com/brocaar/lora-gateway-bridge) for documentation).
2. Build the LoRa server ([click](https://github.com/brocaar/loraserver) for documentation). Use journalctl -u loraserver -f -n 50 to check the status of LoRa server.
3. Build the LoRa app server ([click](https://github.com/brocaar/lora-app-server) for documentation). Use journalctl -u lora-app-server -f -n 50 to check the status of LoRa app server.
4. Use commands of mosquitto\_sub to see the log of the application and the gateway.
5. Power the gateway by a USB charging cable. Connect the LoRa gateway with the PC by a network cable.
6. Observe the parameters on the LED screen of the gateway. Press button S2 to change the parameter version. By default, the server IP is 172.24.0.1. The Up Port and the Down Port are 1700. The IP address of the gateway is 172.24.0.2. These parameters can be changed in minicom by following steps:

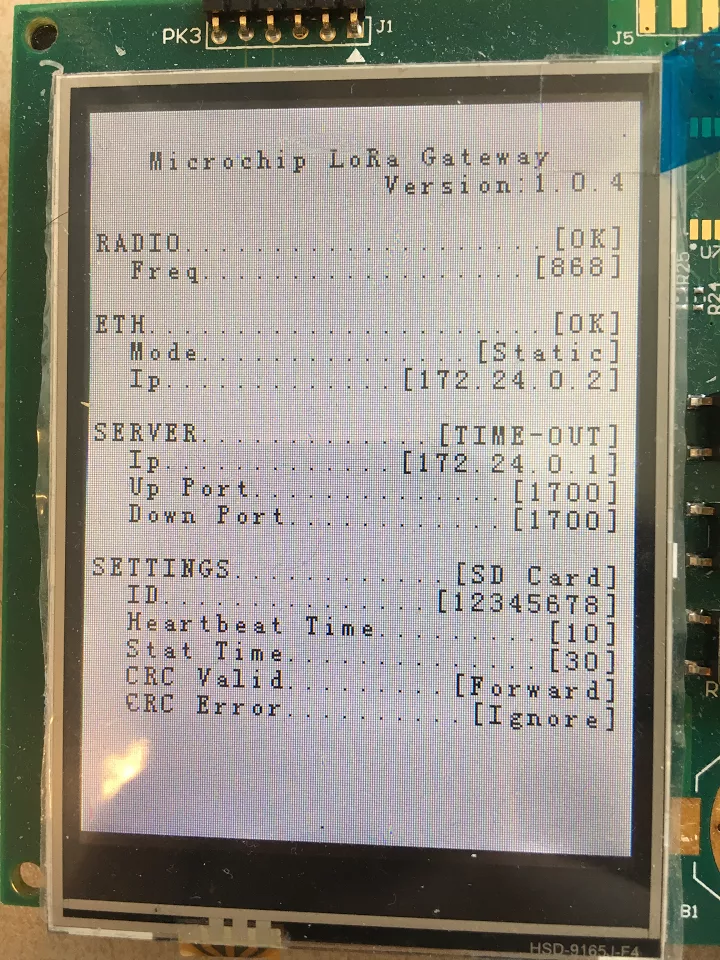


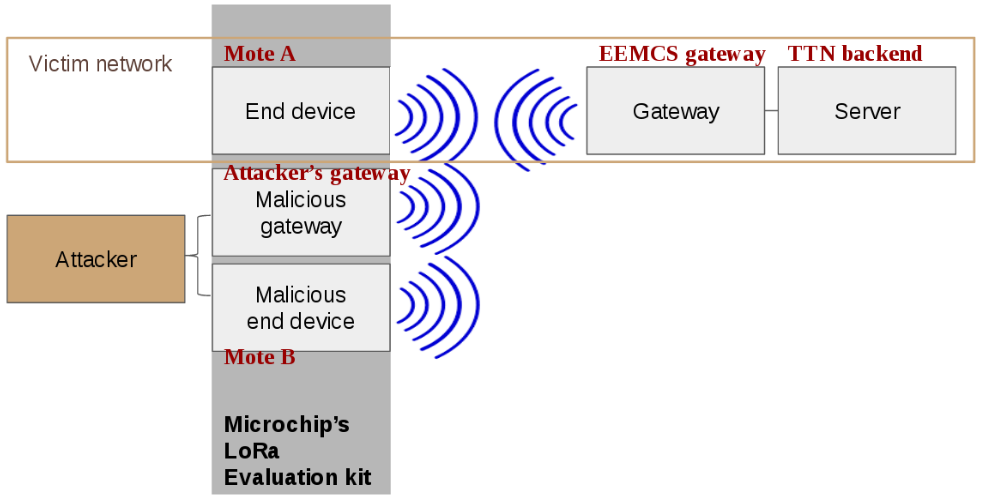
Fig 2. LoRa gateway LED screen before connecting to the server.

* 1. Find the port of LoRa gateway, and open minicom (command: minicom -s)
  2. Check the gateway by using command: sys get ver (Use ctl-M +ctrl-J to end the command). The output of the command should be: Microchip LoRa Gateway Version 1.0.4
  3. Use the command of “sys set svip 198.162.42.105” to set the server IP, and use “sys set svup 1780” and “sys set svdn 1780” to set the up and down ports. Use “sys set ifip 192.168.10.151” to change the gateway’s IP address. Commands for the gateway can be found [here](http://ww1.microchip.com/downloads/en/DeviceDoc/40001827A.pdf).
  4. Use “sys save” to save the configuration in the SD card.

1. Configure the IP address for the LoRa ethernet in the PC. Use the command of “ifconfig eth1 172.24.0.1” to set the server’s IP address. After this configuration, the gateway should be able to communicate with the PC (Use ping to check). On the screen, “TIME-OUT” will become “ONLINE”.
2. Open the browser and use the URL of <https://localhost:8080> to enter the application server. Create an application and a node with parameters of DevEUI, AppEUI, and DevAddr. Choose ABP activation, and set the NwkSKey and AppSKey.
3. Connect a mote to the PC by a USB cable. Use minicom to control the mote. Set the same DevEUI, AppEUI, DevAddr, NwkSkey and AppSKey as the parameters in the application server. Click [here](http://ww1.microchip.com/downloads/en/devicedoc/lora%20mote%20users%20guide.pdf) for mote commands. Use mac save to save the configurations. Use mac join abp to let the mote activated by ABP. Then the mote is able to communicate with the server.
4. There are 2 ways to make the mote send an uplink message to the server.
   1. Use minicom to send mote command. Eg: mac tx uncnf 88 A or mac tx cnf 66 B
   2. Press buttons on the mote. (Use S2 to change the options and S3 to choose)
5. In order to arrange downlink messages, use URL <https://localhost:8080/api> to configure.
6. The server and gateways of The Things Network can also be used. The similar configuration is needed in TTN to set up the communication between the end device and the server.

**Attack 1: Replay attack**

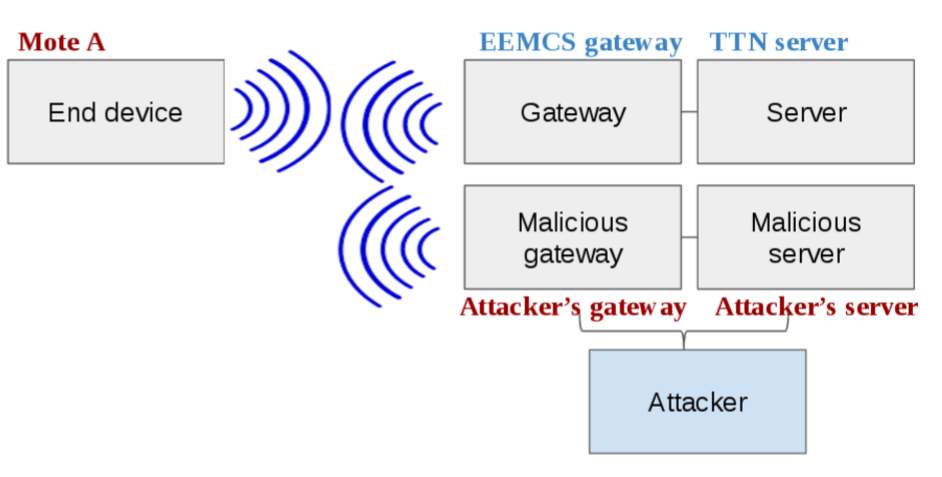
The setup of the replay attack is as follows.



1. The victim's end device is communicating with TTN’s server via the gateway periodically. The end device is activated by ABP manually by pressing the buttons, and it is sending messages periodically. Also, another mote B is connected to the PC.
2. The attacker tunes the gateway to frequency 868 MHz (which is the default value) to listen to any LoRa messages in this frequency band.
3. The attacker uses a script named “replay\_attack.py” to control the malicious gateway to keep capturing messages from Mote A to the TTN gateway. The port number of mote B and devaddr of mote A should be modified in the script.
4. The collected messages are physical payloads in Hex. Devaddr, counter value are in plaintext. These physical payloads are stored by saving the malicious gateway’s log (Use command mosquitto\_sub -t "gateway/#" > filename.txt). The path of saved file should be added in the script.
5. The attacker performs a reset on the victim’s end device. The keys are not changed and the counter values are reset to zero. Then activate the end device by ABP again. Also, the counter value in the victim’s application server should be reset to 0.
6. As long as the malicious gateway observes one reset, the script will automatically find the message, named M, with the same devaddr and the largest counter value in the database. The reset is defined that for the same devaddr, the counter value of the message is smaller than the last one it received.
7. The script will automatically control the malicious mote B as a LoRa radio transmitter to resend the physical payload it just found in database. The message is sent in the same frequency band.
8. If the victim can observe the message M from TTN’s application server, the replay attack is achieved successfully.

**Attack 2: Eavesdropping**

The setup of eavesdropping attack is as follows.



Assumption 1: Plaintext consists of numeric strings

The scrip “cribdragging.py” is used to decrypt messages consist of numeric strings.

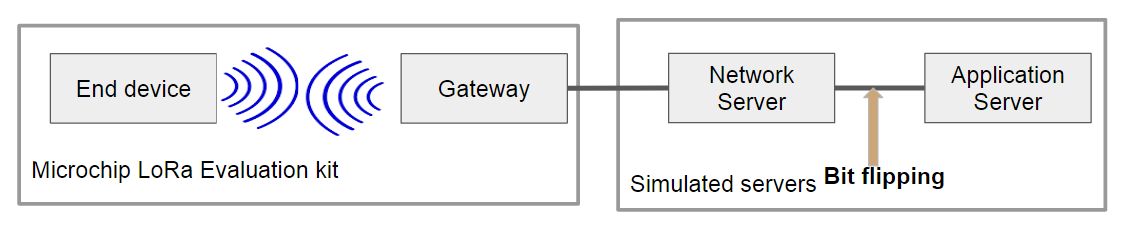
If you only want to see the decryption results, you can use the gateway and application server logs in the folder “cribdragging”. All you need to do is to modify the path in the script and run the script. If you want to collect messages by yourself, then you need to do as follows.

1. The mote A activated by ABP, and it is periodically sending messages (light and temperature value) to the TTN’s gateway and the server. Note that in normal environment, the light value consist of 3 digits. E.g. 556. The malicious gateway is capturing wireless messages. It can read the physical payload of wireless messages from mote A.
2. Save the malicious gateway’s log to a file.
3. Wait till the counter reaches a value c. Reset and reactivate the mote A. Open another new gateway log file and save the log.
4. Change the environment by changing the light to one digit. E.g. 8. Cover the light sensor can help. Make the mote A sending messages to the TTN server from counter value 0. Wait till its counter value reaches c.
5. Repeat step 3 and 4. This time change the light value to 2 digit. E.g. 23
6. Repeat step 3 and 4. This time change the light value to 4 digits. E.g. 1220
7. After communication, we have 4 log files with light value from 1 digit to 4 digits. Define file paths in the script.
8. Run the script. See the description output. Compare the result with the data in the TTN’s server.

Assumption 2: Plaintext consists of character strings.

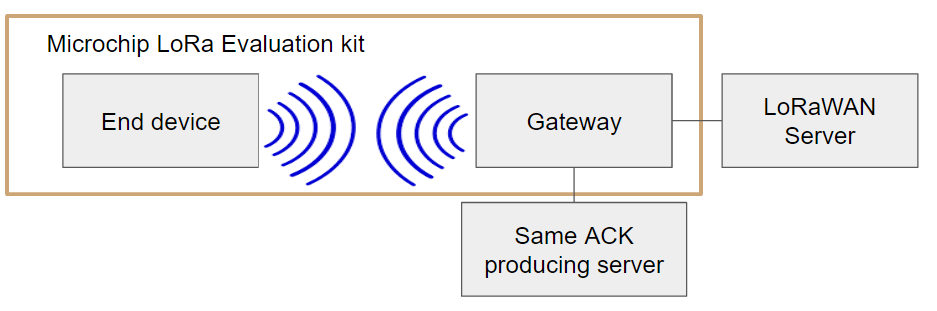
Crib dragging method is used to decrypt alphanumeric-string messages. The attack begins with guessing one possible word, and the derive the corresponding word. The script “cribdragging\_word.py” shows an example of decryption procedure.

**Attack 3: Bit flipping**



1. The servers are simulated in a script named “bitflipping.py”. Before the communication, define the end device’s parameters: DevAddr, NwkSKey and AppSKey in the script.
2. Save the gateway’s log in a file (Use mosquitto\_sub -t "gateway/#" > filename.txt). Define the file path in the script “bitflipping.py”. In this way, the gateway can transfer messages to the network server.
3. Run the script, and let the end device send messages periodically. Every time a new message arrives, the script will flip a bit.
4. The network server has the NwkSKey, and it will calculate the signature and check whether the messages can be accepted or not.

**Attack 4: ACK spoofing**



No script is needed for this attack.

1. Set up the LoRaWAN as above. The server is from Brocaar.
2. Disable the gateway downlink transmission by setting a wrong downlink port number. After setting, the gateway is disabled to send any downlink packets including ACK messages, but it can still receive packets from the LoRaWAN server.
3. Set the end device counter value as: upctr =20, dnctr =10 through minicom.
4. Set the server counter value as: upctr =19, dnctr =11 in the application server (the [website](https://localhost:8080)). Since the upctr value is smaller than the value in the end device, the message from end device with upctr 20 will be accepted.
5. The end device sends a confirmed message M1 to gateway with upctr = 20. The message is accepted in the server and the server will respond an ACK with dnctr = 11.
6. The gateway received the ACK. Since downlink transmission is disabled, no messages are sent to the end device.
7. Since there is no ACK received by the end device, the end device retransmits M1 for 7 times, and then considers M1 to be lost or rejected. For each time of retransmission, the gateway receives a new ACK from the server with dnctr from 11 to 18. After retransmission, the counter value for end device becomes upctr =21, dnctr =10.
8. Enable the gateway downlink transmission, and change the counter value in the application server. Now upctr=20, dnctr= 11. Here the server is work as a same ACK producing server.
9. The end device sends a confirmed message M2 to gateway with upctr = 21, dnctr = 10. The same ACK producing server will reply the ACK with dnctr = 11. And the end device will accept the ACK.