Meshcore authenticated public messages

Issue

On the Cascadia mesh, someone has recently used an AI bot to imitate people, joining in on conversations generally spewing nonsensical BS that looks real. Similarly to traditional signal jamming, the noise created by AI drowns out the signal created by conversation which ultimately results in people getting exhausted and leaving the conversation.

MeshCore public group messages are completely unauthenticated. Both users and most repeater owners have an expectation that the protocol is secure regardless of whether messages are being sent privately or publicly. What's a nuisance today could be actively dangerous in a situation where the mesh is actually needed for emergency communications.

Currently there seems to be a desire to track down the individual(s) responsible. This is a pointless exercise, a $20 Heltec and ChatGPT is all that's required to actually cause this type of disruption. If we catch someone today, tomorrow someone else will be exploiting the same vulnerability, and the mesh will soon die as people lose interest.

Some tough pills to swallow

1. The longer we wait to solve this problem, the more difficult it will become. Today is the cheapest it's ever going to be.
2. MeshCore's protocol is fundamentally broken.
    a. Fixing it will result in legacy clients and repeaters not working with new traffic.
    b. Leaving it in this broken state will result in people giving up and moving to a protocol that doesn't have these problems.
3. It's impossible to have a Mesh open to everyone while also blocking it from people who abuse the system.
4. Tradeoffs will need to be made
    a. A LORA packet is 255 bytes.
    b. An ED25519 signature is 64 bytes bytes, no exception
    c. An ED25519 public key is 32 bytes. Vanity keys can reduce this but these have to be bruteforced which increases expenses.

Solution

1. All flood traffic must be authenticated with an ED25519 signature. (~96 byte cost per message)

2. The trustworthiness of an ED25519 public key is determined by the number of 0's as its suffix. These are vanity keys that require bruteforcing to calculate a valid public/private key pair that match this criteria.
    a. A single zero suffix is instant. (32 byte key length)
    b. Minting a key that ends with 7 zero suffix is 5.37 seconds. (29 byte key length).
    c. Minting a key that ends with 8 zero suffix is 1.43 minutes (28 byte key length).
    d. Minting a key that ends with 9 zero suffix is 22 minutes (28 byte key length) .
    e. Minting a key that ends with 10 zeros takes 6 hours. (27 byte key length).
    f. Minting a key that ends with 11 zeros takes 4 days (27 byte key length).
    g. Minting a key that ends with 12 zeros takes 65 days. (26 byte key length).
3. Repeaters and companions can filter lowly trusted traffic, and hop counts potentially reduced in the event of trouble being caused on the network.
4. The current 64 byte repeater array and 4 byte flood scope will be removed and replaced with a 13 byte bloom filter.
    a. A bloom filter acts as a set, we can test for the presence of a repeater in the set (and know with certainty whether it is not contained, and probabilistically whether it is contained).
    b. We lose the ordering of repeaters that the array provides, but gain the ability to combine bloom filters to create routes that contain multiple potential repeaters. Traffic will likely reduce as nodes no longer have to flood and rediscover a new path to send a message just as the old path starts working again.
    c. For flood scopes, you merely insert the flood scope into the bloom filter. Repeater does a contains check and forwards if the region is present, or else checks for "ANY_SCOPE" for unscoped traffic.
    d. False positive rate for a 13 byte bloom filter is approximately 4.46% if it contains 15 hops. The odds of that false positive actually hitting is significantly lower, as it requires a neighbor along the route to match the false positive value.
5. Plaintext group messages are reduced to **139 bytes** per message, down from **160 bytes.** This is the real cost.
    a. We could remove the name from the message being sent given that it's already a signed message.
    b. We could start huffman encoding messages (see analysis here), which will likely give an additional 35% of usable data, bringing the message limit up to 180 bytes.
6. Repeaters can carry on repeating legacy traffic. Companions can choose to filter legacy traffic from public channels.

Packet format comparison

Original meshcore Group Message

Field	Size	Data
Meshcore Header	1	Version 0 Payload type=GRP_TXT. Route type=FLOOD
Transport code 1&2	4	Region/scope filtering
Path metadata	1	Size of path array
Path array	64	Each repeater
Channel Hash	1	
Channel HMAC	2	
AES ciphertext	176	User message

New Meshcore Group Message

Field	Size	Data
Meshcore Header	1	Version 3 (experimental) Payload type=GRP_TXT Route type=FLOOD
Hop count	1	Number of hops since sent (previously calculated with len(repeaterArray))
Combined scope/route bloom filter	13	
Channel Hash	1	
Channel HMAC	2	
ED25519 public key	29	Assumes vanity with 7 zero suffix. 5 second generation time
ED25519 signature	64	Signed over the AES encrypted data, not plaintext.
AES ciphertext	144	

1. All other packet types containing scope and a repeater array will be replaced with a bloom filter for consistency.
2. Adverts already contain public keys and signatures, their format doesn't have to be changed further.
3. For privacy, we can keep public keys & signatures away from direct private messages. That traffic is already authenticated.