44 - Encryption

Jump to bottom

Nunu edited this page on Feb 12 · 18 revisions

Encryption algorithm

This firmware uses ChaCha20 as the encryption algorithm (based on the library written by @grigorig https://github.com/grigorig/chachapoly).

Encryption key

The encryption key used has 256 bits.

It is derived from the 10 characters password (80 bits) set by the user in the `EncKey` menu item and is stored in the EEPROM memory of the device.

The key derivation function provides 256 bit encryption key by combining 4 x 64 bit hashes of the same (80 bits) password. Each hash is generated with a different 64 bit salt. The encryption key is stored in the RAM memory of the device and is lost once the device is powered down.

So the actual encryption key is never stored permanently in the EEPROM memory, instead it is calculated when needed.

Of course if the EEPROM password is read, it can be copied to other devices and these devices will derive it to the same encryption key, which is actually desired as it allows setting up multiple encrypted radios in bulk via CHIRP.

Users can further improve security and prevent the EEPROM password to be read via serial connection, by setting `Passwd` option in the menu. This way serial connection is prevented until correct password is entered on boot.

The device generates a second short 8 ASCII chars hash from the (80 bits) password, but with different display salt so users can compare these hashes and determine if they are both using the same key. It is not used for an actual encryption.

Random number generator

Random numbers are used to generate a different, random nonce with each invocation of the of the encryption algorithm. This makes each message look different to third parties, even if the message contains the same plaintext content.

Currently random numbers are generated by reading the least significant bits of the noise register `BK4819_REG_65` from the currently used frequency.

This comes with some security caveats, such as the fact that third parties may generate a strong signal on our frequency, affecting the random numbers we generate. Based on this, in the future, and if the binary size allows, this algorithm will be extended to:

- XOR the noise reading from the BK4819 and the RSSI reading from the FM Radio chip (of the currently listening frequency),
- condition the random sequence so each random byte must be different by 10% from the previous one.

Hash function

The hash operations are performed using a 64 bit Fowler–Noll–Vo hash function.

Operation

Only packet payload is being encrypted because it is the only data generated by a human, so it doesn't have a known value or structure the third-party could know or predict easily, thus making it harder to decrypt.

This is also the reason why the `ACK` messages are not decrypted, encrypted or even authenticated.

The radio will never reply with encrypted received data in any form, as third-party could inject plain-text and collect encrypted data also known as the chosen plaintext attack.

- Encryption key
- Random number generator
- Hash function
- Operation

- **50 - FM broadcast radio receiver**
- **60 - Flashing the firmware**
- **70 - Differences from egzumer firmware**
- **80 - Tips & Tricks and Known Issues**
- **90 - 00 - Early Release V.21.x - Features**
- **90 - 20 - Menu**
- **99 - Updates on this Manual**

Clone this wiki locally

```
https://github.com/kamilsss655/uv-k5-firmware-custom.wiki.git
```