

Examples:

Assumptions:

InputStick is password protected. Key is set to „abc123”

Keyboard layout used by USB host: en-US.

Initialization procedure:

1. Send *CMD* 0x04 „Run firmware”.

It is possible that bootloader application can be still running when you connect to InputStick. After sending this command you can be sure that main firmware is launched.

2. Send *CMD* 0x10 „Get firmware info”.

This step allows you to learn about firmware version (it is possible that firmware was upgraded since last time) and whether password protection is enabled.

3. (Optional) Send *CMD* 0x12 „Authenticate”.

This step is necessary when InputStick is password protected.

4. Wait for *CMD* 0x2F „Status update”.

Make sure that USB host set InputStick state to configured (Ready) and HID buffers are empty. This command will be sent by InputStick approximately every 100 ms.

Send HID reports (Keyboard / Mouse / Consumer control).

Continue receiving status updates.

~~BUG (v0.97): when CRC32 is set to 0x00, 0x00, 0x00, 0x00, InputStick will send error notification (CMD: 0x1F, error code: 0xE2). Command will be accepted and executed anyway. (fixed in v0.98)~~

Send CMD 0x04 „Run firmware”.

- 1) Reserve 4 bytes for CRC32 value:

0x00	0x00	0x00	0x00												
------	------	------	------	--	--	--	--	--	--	--	--	--	--	--	--

- 2) Add *CMD*:

CMD = 0x04, „Run firmware”.

0x00	0x00	0x00	0x00	0x04											
------	------	------	------	------	--	--	--	--	--	--	--	--	--	--	--

- 3) Add *PARAM* and *DATA*:

PARAM and *DATA* are not specified for this command. In such case set value of *PARAM* to 0x00 and do not add any *DATA* bytes.

0x00	0x00	0x00	0x00	0x04	0x00										
------	------	------	------	------	------	--	--	--	--	--	--	--	--	--	--

- 4) Add padding:

Use “0x00”s. Total length must be a multiple of 16 (16, 32, 48 ...).

0x00	0x00	0x00	0x00	0x04	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

- 5) Calculate CRC32:

0x78	0xD3	0xFD	0x10	0x04	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

Do not include first 4 bytes in CRC32 calculation. Padding bytes must be included. Store CRC32 value in first 4 bytes (reserved in step 1.). You can skip this step and leave all 4 bytes as 0x00. In such case InputStick will not perform CRC32 check.

Example (web calculator: <http://www.fileformat.info/tool/hash.htm>):

Binary hash, hex bytes: 0x04 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
CRC32: 78d3fd10

Example (Java, Android):

```
CRC32 mCrc = new CRC32();  
byte[] packet = new byte[16];  
packet[4] = 0x04;  
mCrc.reset();  
mCrc.update(packet, CRC_OFFSET, packet.length - CRC_OFFSET); //CRC_OFFSET = 4  
long crcValue = mCrc.getValue();  
packet[3] = (byte)crcValue;  
crcValue >>= 8;  
packet[2] = (byte)crcValue;  
crcValue >>= 8;
```

```
packet[1] = (byte)crcValue;
crcValue >>= 8;
packet[0] = (byte)crcValue;
```

6) Encrypt the packet:

There is no need to encrypt this packet (no sensitive information is transferred) Also, since *Authenticate CMD* was not yet executed, it is not possible to use encryption anyway.

7) Create header:

First header byte must be always set to 0x55 (*START_TAG*):

0x55	
------	--

Payload length is 16 bytes: 16 x 1. Set value of length to 1: 0x01.
Set *RESP* flag (0x80), otherwise InputStick will not provide response.
Payload is not encrypted. Do NOT set *ENCR* flag (0x40).

INFO_BYTE: 0x01 | 0x80 | 0x00 = 0x81

0x55	0x81
------	------

8) Send header and follow with payload:

0x55	0x81
------	------

0x78	0xD3	0xFD	0x10	0x04	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

Important:

After sending packet with *RESP* flag SET, you cannot send another packet with *RESP* flag SET before receiving packet with *RESP* flag SET (response).

You can send packets with *RESP* flag NOT SET.

Receive response.

- 1) Wait until *START_TAG* is received:

0x55	
------	--

- 2) Read *INFO_BYTE*:

0x55	0x81
------	------

INFO_BYTE:

RESP flag is SET: $0x81 \& 0x80 = 0x80 \neq 0$

ENCR flag is NOT SET: $0x81 \& 0x40 = 0x00 = 0$

Payload length is 16 bytes. $0x81 \& 0x3F = 0x01 = 1$, $1 \times 16 = 16$

- 3) Receive payload data:

Read 16 bytes.

0xB9	0x5D	0x22	0xD0	0x04	0x01	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

- 4) If *ENCR* flag is SET, decrypt the packet:

ENCR flag is NOT SET, skip this step.

- 5) Verify CRC32:

If payload is not encrypted, this step can be skipped. Bluetooth already takes care of maintaining data integrity. In case of encrypted packet, CRC32 verification allows to learn if decryption process was successful.

Get CRC32 value from first 4 bytes:

CRC32 = 0xB95D22D0

Calculate CRC32, skip first 4 bytes:

CRC32(0x04, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00) = b95d22d0

Check if both values are identical. If not, reject the packet.

- 6) Identify *CMD*:

0xB9	0x5D	0x22	0xD0	0x04	0x01	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

0x04 = „Run firmware“. This packet is a response to previously sent „Run firmware“ *CMD*.

- 7) If *RESP* flag is SET, check value of *RESP_CODE*:

0xB9	0x5D	0x22	0xD0	0x04	0x01	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

CMD 0x04 always executes successfully and returns 0x01.

8) Get *DATA* (*RESP* flag NOT SET) / *RESP_DATA* (*RESP* flag SET):

There is not *RESP_DATA* specified for *CMD* 0x04.

Important:

After receiving packet with *RESP* flag SET, it is now possible to send another packet with *RESP* flag SET.

Send **CMD 0x10** „Get firmware info“.

1) Reserve 4 bytes for CRC32 value:

0x00	0x00	0x00	0x00												
------	------	------	------	--	--	--	--	--	--	--	--	--	--	--	--

2) Add **CMD**:

0x00	0x00	0x00	0x00	0x10											
------	------	------	------	------	--	--	--	--	--	--	--	--	--	--	--

3) Add **PARAM and DATA**:

0x00	0x00	0x00	0x00	0x10	0x00										
------	------	------	------	------	------	--	--	--	--	--	--	--	--	--	--

4) Add padding:

0x00	0x00	0x00	0x00	0x10	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

5) Calculate CRC32:

0x77	0xCD	0x2B	0x93	0x10	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

6) Encrypt the packet:

7) Create header:

0x55	0x81
------	------

Payload length is 16 bytes: 16 x 1. Set value of length to 1: 0x01.
Set **RESP** flag (0x80), otherwise InputStick will not provide response.
Payload is not encrypted. Do NOT set **ENCR** flag (0x40).

INFO_BYTE: 0x01 | 0x80 | 0x00 = 0x81

8) Send header and follow with payload:

0x55	0x81
------	------

0x77	0xCD	0x2B	0x93	0x10	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

Receive response.

- 1) Wait until *START_TAG* is received:

0x55	
------	--

- 2) Read *INFO_BYTE*:

0x55	0x83
------	------

INFO_BYTE:

RESP flag is SET: $0x83 \& 0x80 = 0x80 \neq 0$

ENCR flag is NOT SET: $0x83 \& 0x40 = 0x00 = 0$

Payload length is 48 bytes. $0x83 \& 0x3F = 0x03 = 3, 3 \times 16 = 48$

- 3) Receive payload data:

0xDF	0xAB	0x7A	0x09	0x10	0x01	0x42	0x00	0x5F	0x01	0x00	0x00	0x01	0x05	0x08	0x00
0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x01	0x00	0x00	0x00	0x00	0x00	0x00	0x00
0x21	0x34	0x00	0x00	0x00	0x24	0x00	0x00	0x00	0x24	0x00	0x00	0x00	0x00	0x00	0x00

- 4) If *ENCR* flag is SET, decrypt the packet:

- 5) Verify CRC32:

CRC32 = 0xDFAB7A09

Calculate CRC32, skip first 4 bytes:

CRC32(0x10, 0x01, ..., 0x24, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00) = 0xdfab7a09

- 6) Identify *CMD*:

0xDF	0xAB	0x7A	0x09	0x10	0x01	0x42	0x00	0x5F	0x01	0x00	0x00	0x01	0x05	0x08	0x00
0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x01	0x00	0x00	0x00	0x00	0x00	0x00	0x00
0x21	0x34	0x00	0x00	0x00	0x24	0x00	0x00	0x00	0x24	0x00	0x00	0x00	0x00	0x00	0x00

- 7) If *RESP* flag is SET, check value of *RESP_CODE*:

0xDF	0xAB	0x7A	0x09	0x10	0x01	0x42	0x00	0x5F	0x01	0x00	0x00	0x01	0x05	0x08	0x00
0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x01	0x00	0x00	0x00	0x00	0x00	0x00	0x00
0x21	0x34	0x00	0x00	0x00	0x24	0x00	0x00	0x00	0x24	0x00	0x00	0x00	0x00	0x00	0x00

CMD 0x10 always executes successfully and returns 0x01.

- 8) Get *DATA* (*RESP* flag NOT SET) / *RESP_DATA* (*RESP* flag SET):

0xDF	0xAB	0x7A	0x09	0x10	0x01	0x42	0x00	0x5F	0x01	0x00	0x00	0x01	0x05	0x08	0x00
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x01	0x00	0x00	0x00	0x00	0x00	0x00	0x00
0x21	0x34	0x00	0x00	0x00	0x24	0x00	0x00	0x00	0x24	0x00	0x00	0x00	0x00	0x00	0x00

Bytes:	Value:	Description
0	0x42	Firmware type: „B”
1..2	0x00, 0x5F	Firmware version: 0.95
3	0x01	Hardware revision: 0x01
4..5	0x00, 0x00	Reserved
6	0x01	USB pullup: ENABLED
7	0x05	USB state: READY (Configured)
8	0x08	Startup status: OK
9..15	0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00	Reserved
16	0x00	Encrypt outgoing data: DISABLED
17	0x00	Authenticated: NO
18	0x01	Password protection: ENABLED
19..23	0x00, 0x00, 0x00, 0x00, 0x00	Reserved
24..27	0x00, 0x00, 0x21, 0x34	System uptime: 0x00002134 = 8500 ms = 0min, 08s, 500ms
28..31	0x00, 0x00, 0x00, 0x24	Total received (RX) bytes: 0x00000024 = 36B
32..35	0x00, 0x00, 0x00, 0x24	Total sent (TX) bytes: 0x00000024 = 36B

Send CMD 0x12 „Authenticate”.

If InputStick is password protected, you must successfully go through authentication process, before you can send any HID reports.

In previous step, data received from InputStick revealed that password protection is enabled. It is now necessary to send „Authenticate” *CMD*. Otherwise this step should be skipped.

Encryption key must be provided by user. Assume that password is set to „abc123”, to get AES-128 key, use MD5(password):

key = MD5(“abc123”) = e99a18c428cb38d5f260853678922e03

1) Reserve 4 bytes for CRC32 value:

0x00	0x00	0x00	0x00												
------	------	------	------	--	--	--	--	--	--	--	--	--	--	--	--

2) Add *CMD*:

0x00	0x00	0x00	0x00	0x12											
------	------	------	------	------	--	--	--	--	--	--	--	--	--	--	--

3) Add *PARAM* and *DATA*:

0x00	0x00	0x00	0x00	0x12	0x01										
------	------	------	------	------	------	--	--	--	--	--	--	--	--	--	--

PARAM = 0x01 – allow to encrypt outgoing packets.

Generate and add *AES-128 IV* (Initialize Vector):

0x00	0x00	0x00	0x00	0x12	0x01	0x87	0x27	0x46	0x3E	0x91	0x05	0x6D	0x7A	0xA8	0x5C
0x66	0x32	0xFF	0x7C	0x30	0xCB										

Example (Java, Android):

```
Cipher mCipherEncr;  
Cipher mCipherDecr;  
SecretKeySpec mKey;  
byte[] iv = null;  
mKey = new SecretKeySpec(key, "AES");  
mCipherEncr = Cipher.getInstance("AES/CBC/NoPadding");  
mCipherEncr.init(Cipher.ENCRYPT_MODE, mKey);  
iv = mCipherEncr.getIV();
```

Generate and add *KeyVerificationData*:

0x00	0x00	0x00	0x00	0x12	0x01	0x87	0x27	0x46	0x3E	0x91	0x05	0x6D	0x7A	0xA8	0x5C
0x66	0x32	0xFF	0x7C	0x30	0xCB	0x79	0x0E	0xBE	0x67	0x11	0x3D	0xE9	0x5D	0x62	0x42
0x09	0xE0	0xD2	0xA9	0xF9	0xB3										

Example (Java, Android):

```
byte[] tmp = new byte[16];
r.nextBytes(tmp);
mCrc.reset();
mCrc.update(tmp, 4, 12); //only 12 bytes!
long crcValue = mCrc.getValue();
tmp[3] = (byte)crcValue;
crcValue >>= 8;
tmp[2] = (byte)crcValue;
crcValue >>= 8;
tmp[1] = (byte)crcValue;
crcValue >>= 8;
tmp[0] = (byte)crcValue;
tmp = mCipherEncr.update(tmp);
```

Generate and add *KeyChallengeData*:

0x00	0x00	0x00	0x00	0x12	0x01	0x87	0x27	0x46	0x3E	0x91	0x05	0x6D	0x7A	0xA8	0x5C
0x66	0x32	0xFF	0x7C	0x30	0xCB	0x79	0x0E	0xBE	0x67	0x11	0x3D	0xE9	0x5D	0x62	0x42
0x09	0xE0	0xD2	0xA9	0xF9	0xB3	0xBF	0xC4	0x37	0x7D	0x45	0xD2	0xED	0xD5	0x2F	0xA7
0x01	0xE6	0x6C	0x66	0x90	0xAB										

Example (Java, Android):

```
challengeData = new byte[16];
r.nextBytes(challengeData);
```

4) Add padding:

0x00	0x00	0x00	0x00	0x12	0x01	0x87	0x27	0x46	0x3E	0x91	0x05	0x6D	0x7A	0xA8	0x5C
0x66	0x32	0xFF	0x7C	0x30	0xCB	0x79	0x0E	0xBE	0x67	0x11	0x3D	0xE9	0x5D	0x62	0x42
0x09	0xE0	0xD2	0xA9	0xF9	0xB3	0xBF	0xC4	0x37	0x7D	0x45	0xD2	0xED	0xD5	0x2F	0xA7
0x01	0xE6	0x6C	0x66	0x90	0xAB	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

5) Calculate CRC32:

0xC6	0x15	0x44	0x67	0x12	0x01	0x87	0x27	0x46	0x3E	0x91	0x05	0x6D	0x7A	0xA8	0x5C
0x66	0x32	0xFF	0x7C	0x30	0xCB	0x79	0x0E	0xBE	0x67	0x11	0x3D	0xE9	0x5D	0x62	0x42
0x09	0xE0	0xD2	0xA9	0xF9	0xB3	0xBF	0xC4	0x37	0x7D	0x45	0xD2	0xED	0xD5	0x2F	0xA7
0x01	0xE6	0x6C	0x66	0x90	0xAB	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

6) Encrypt the packet:

Encryption is not enabled, key has not been verified yet.

7) Create header:

0x55	0x84
------	------

Payload length is 64 bytes: 16 x 4. Set value of length to 1: 0x04.
Set *RESP* flag (0x80), otherwise InputStick will not provide response.
Payload is not encrypted. Do NOT set *ENCR* flag (0x40).

INFO_BYTE: 0x04 | 0x80 | 0x00 = 0x84

8) Send header and follow with payload:

0x55	0x84
------	------

0xC6	0x15	0x44	0x67	0x12	0x01	0x87	0x27	0x46	0x3E	0x91	0x05	0x6D	0x7A	0xA8	0x5C
0x66	0x32	0xFF	0x7C	0x30	0xCB	0x79	0x0E	0xBE	0x67	0x11	0x3D	0xE9	0x5D	0x62	0x42
0x09	0xE0	0xD2	0xA9	0xF9	0xB3	0xBF	0xC4	0x37	0x7D	0x45	0xD2	0xED	0xD5	0x2F	0xA7
0x01	0xE6	0x6C	0x66	0x90	0xAB	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

Receive response.

- 1) Wait until *START_TAG* is received:

0x55	
------	--

- 2) Read *INFO_BYTE*:

0x55	0x82
------	------

INFO_BYTE:

RESP flag is SET: $0x82 \& 0x80 = 0x80 \neq 0$

ENCR flag is NOT SET: $0x82 \& 0x40 = 0x00 = 0$

Payload length is 32 bytes. $0x82 \& 0x3F = 0x02 = 2, 2 \times 16 = 32$

- 3) Receive payload data:

0x29	0x5B	0x75	0x7F	0x12	0x01	0xFB	0x7B	0x7D	0x9B	0xF5	0x1B	0xB6	0x47	0x57	0xA2
0xC4	0x8E	0x28	0xFA	0x97	0xD5	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

- 4) If *ENCR* flag is SET, decrypt the packet:

- 5) Verify CRC32:

CRC32 = 0x295B757F

Calculate CRC32, skip first 4 bytes:

CRC32(0x12, 0x01 ..., 0x00) = 0x295b757f

- 6) Identify *CMD*:

0x29	0x5B	0x75	0x7F	0x12	0x01	0xFB	0x7B	0x7D	0x9B	0xF5	0x1B	0xB6	0x47	0x57	0xA2
0xC4	0x8E	0x28	0xFA	0x97	0xD5	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

- 7) If *RESP* flag is SET, check value of *RESP_CODE*:

0x29	0x5B	0x75	0x7F	0x12	0x01	0xFB	0x7B	0x7D	0x9B	0xF5	0x1B	0xB6	0x47	0x57	0xA2
0xC4	0x8E	0x28	0xFA	0x97	0xD5	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

0x01 – OK (authentication was successful).

0x20 – Invalid encryption key.

0x21 – Password protection is not enabled.

- 8) Get *DATA* (*RESP* flag NOT SET) / *RESP_DATA* (*RESP* flag SET):

0x29	0x5B	0x75	0x7F	0x12	0x01	0xFB	0x7B	0x7D	0x9B	0xF5	0x1B	0xB6	0x47	0x57	0xA2
0xC4	0x8E	0x28	0xFA	0x97	0xD5	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

KeyResponseData = 0xFB ... 0xD5 (16 bytes).

Decrypt *KeyResponseData*, and compare result with previously sent *KeyChallengeData*.

Example (Java, Android):

```
byte[] tmp = mAes.decrypt(responseData);  
if (Arrays.equals(tmp, challengeData)) {...}
```

Wait for **CMD 0x2F** „Status update“.

- 1) Wait until *START_TAG* is received:

0x55	
------	--

- 2) Read *INFO_BYTE*:

0x55	0x01
------	------

INFO_BYTE:

RESP flag is NOT SET: $0x01 \& 0x80 = 0x00 = 0$

ENCR flag is NOT SET: $0x01 \& 0x40 = 0x00 = 0$

Payload length is 16 bytes. $0x01 \& 0x3F = 0x01 = 1, 1 \times 16 = 16$

- 3) Receive payload data:

0x20	0x96	0xEF	0x11	0x2F	0x05	0x05	0x01	0x01	0x01	0x01	0x01	0x01	0x00	0x00	0x00	0xFF
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

- 4) If *ENCR* flag is SET, decrypt the packet:

- 5) Verify CRC32:

CRC32 = 0x2096EF11

Calculate CRC32, skip first 4 bytes:

CRC32(0x2F, 0x05 ..., 0xFF) = 0x2096ef11

- 6) Identify *CMD*:

0x20	0x96	0xEF	0x11	0x2F	0x05	0x05	0x01	0x01	0x01	0x01	0x01	0x01	0x00	0x00	0x00	0xFF
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

- 7) If *RESP* flag is SET, check value of *RESP_CODE*:

RESP flag is NOT SET, there is no *RESP_CODE*.

- 8) Get *DATA* (*RESP* flag NOT SET) / *RESP_DATA* (*RESP* flag SET):

RESP flag is NOT SET. *DATA* starts immediately after *CMD* byte, there is not *RESP_CODE* byte.

0x20	0x96	0xEF	0x11	0x2F	0x05	0x05	0x01	0x01	0x01	0x01	0x01	0x01	0x00	0x00	0x00	0xFF
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

Bytes:	Value:	Description
0	0x05	Device state: READY
1	0x05	Keyboard LEDs: NumLock, ScrollLock
2	0x01	Keyboard protocol: Report Protocol

3	0x01	Keyboard buffer empty: Yes
4	0x01	Mouse protocol: Report Protocol
5	0x01	Mouse buffer empty: Yes
6	0x01	Consumer buffer empty: Yes
7	0x00	Keyboard reports sent to host: 0
8	0x00	Mouse reports sent to host: 0
9	0x00	Consumer reports sent to host: 0
10	0xFF	Reserved (0xFF)

Important:

RESP flag is NOT SET. It is possible that this command is received between sending packet with *RESP* flag SET and receiving response packet with *RESP* flag SET. It should not trigger sending another packet with *RESP* flag SET.

Send HID keyboard report.

1) Reserve 4 bytes for CRC32 value:

0x00	0x00	0x00	0x00												
------	------	------	------	--	--	--	--	--	--	--	--	--	--	--	--

2) Add *CMD*:

0x00	0x00	0x00	0x00	0x21											
------	------	------	------	------	--	--	--	--	--	--	--	--	--	--	--

3) Add *PARAM* and *DATA*:

Example: type „A”:

Press left Shift key (0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00).

Hold left Shift key, press „A” key (0x02, 0x00, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00).

Release all keys (0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00).

Add *PARAM* (number of HID keyboard reports):

0x00	0x00	0x00	0x00	0x21	0x03										

Add *DATA* (HID reports):

0x00	0x00	0x00	0x00	0x21	0x03	0x02	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x02	0x00
0x04	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00		

4) Add padding:

0x00	0x00	0x00	0x00	0x21	0x03	0x02	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x02	0x00
0x04	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

5) Calculate CRC32:

0xC0	0x41	0x68	0xFC	0x21	0x03	0x02	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x02	0x00
0x04	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

6) Encrypt the packet:

7) Create header:

0x55	0x02
------	------

Payload length is 32 bytes: 16 x 2. Set value of length to 2: 0x02.

Do NOT SET *RESP* flag (0x80), *RESP_CODE* can be ignored, no *RESP_DATA* will be provided.

Payload is not encrypted. Do NOT set *ENCR* flag (0x40).

INFO_BYTE: 0x02 | 0x00 | 0x00 = 0x02

8) Send header and follow with payload:

0x55	0x02
-------------	-------------

0xC0	0x41	0x68	0xFC	0x21	0x03	0x02	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x02	0x00
0x04	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

Important:

After sending this CMD, you should assume that there are 3 HID reports currently being stored in keyboard buffer. Remaining capacity: $32 - 3 = 29$ HID keyboard reports.

After some time InputStick will provide another status update:

0x32	0x23	0x40	0xFF	0x2F	0x05	0x05	0x01	0x01	0x01	0x01	0x01	0x03	0x00	0x00	0xFF
------	------	------	------	------	------	------	------	------	------	------	------	-------------	------	------	------

Keyboard reports sent to host: 3

At this moment you can increase available buffer space by 3: $29 + 3 = 32$, assuming that no HID reports were sent to buffer in-between.

Send HID keyboard report (Encrypted).

1) Reserve 4 bytes for CRC32 value:

0x00	0x00	0x00	0x00												
------	------	------	------	--	--	--	--	--	--	--	--	--	--	--	--

2) Add *CMD*:

0x00	0x00	0x00	0x00	0x21											
------	------	------	------	------	--	--	--	--	--	--	--	--	--	--	--

3) Add *PARAM* and *DATA*:

Example: type „a”:

All keys released (no modifier key is required) (0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00).

Hold left Shift key, press „A” key (0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00).

Release all keys (0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00).

Add *PARAM* (number of HID keyboard reports):

0x00	0x00	0x00	0x00	0x21	0x03										

Add *DATA* (HID reports):

0x00	0x00	0x00	0x00	0x21	0x03	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
0x04	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00		

4) Add padding:

0x00	0x00	0x00	0x00	0x21	0x03	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
0x04	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

5) Calculate CRC32:

0xD0	0xDF	0x48	0x10	0x21	0x03	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
0x04	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00

6) Encrypt the packet:

0x50	0xE4	0x01	0x76	0xE1	0x5D	0x0E	0x35	0x6D	0x5A	0xA8	0xD8	0xF8	0x9F	0x98	0xB7
0x84	0x3D	0x66	0xD2	0x41	0x73	0xF3	0x97	0xE2	0x25	0x50	0xA2	0xF8	0x1F	0xDA	0xBD

7) Create header:

0x55	0x42
------	------

Payload length is 32 bytes: 16 x 2. Set value of length to 2: 0x02.

Do NOT SET *RESP* flag (0x80), *RESP_CODE* can be ignored, no *RESP_DATA* will be provided.

Payload IS encrypted. Set *ENCR* flag (0x40).

INFO_BYTE: 0x02 | 0x00 | 0x40 = 0x42

8) Send header and follow with payload:

0x55	0x42
------	------

0x50	0xE4	0x01	0x76	0xE1	0x5D	0x0E	0x35	0x6D	0x5A	0xA8	0xD8	0xF8	0x9F	0x98	0xB7
0x84	0x3D	0x66	0xD2	0x41	0x73	0xF3	0x97	0xE2	0x25	0x50	0xA2	0xF8	0x1F	0xDA	0xBD

Send encrypted *CMD* 0x11.

1) Reserve 4 bytes for CRC32 value:

0x00	0x00	0x00	0x00												
------	------	------	------	--	--	--	--	--	--	--	--	--	--	--	--

2) Add *CMD*:

0x00	0x00	0x00	0x00	0x11											
------	------	------	------	------	--	--	--	--	--	--	--	--	--	--	--

3) Add *PARAM* and *DATA*:

0x00	0x00	0x00	0x00	0x11	0x00										

4) Add padding:

0x00	0x00	0x00	0x00	0x11	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

5) Calculate CRC32:

0xEC	0x68	0x67	0xFC	0x11	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

6) Encrypt the packet:

0x14	0x3D	0xD0	0xE9	0xCD	0x62	0xDD	0x13	0x72	0x70	0x23	0x3E	0x32	0x70	0xBA	0x86
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

7) Create header:

0x55	0xC1
------	------

Payload length is 16 bytes: 16 x 1. Set value of length to 1: 0x01.

SET *RESP* flag (0x80).

Payload IS encrypted. Set *ENCR* flag (0x40).

INFO_BYTE: 0x01 | 0x80 | 0x40 = 0xC1

8) Send header and follow with payload:

0x55	0xC1
------	------

0x14	0x3D	0xD0	0xE9	0xCD	0x62	0xDD	0x13	0x72	0x70	0x23	0x3E	0x32	0x70	0xBA	0x86
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

Receive encrypted response.

- 1) Wait until *START_TAG* is received:

0x55	
------	--

- 2) Read *INFO_BYTE*:

0x55	0xC1
------	------

INFO_BYTE:

RESP flag is SET: $0xC1 \& 0x80 = 0x80 \neq 0$

ENCR flag is SET: $0xC1 \& 0x40 = 0x40 \neq 0$

Payload length is 16 bytes. $0xC1 \& 0x3F = 0x01 = 1$, $1 \times 16 = 16$

- 3) Receive payload data:

0x1E	0x20	0x75	0x13	0xFD	0x01	0x3F	0x0C	0xA4	0x1E	0xE6	0x71	0x0B	0x68	0xC4	0x4F
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

- 4) If *ENCR* flag is SET, decrypt the packet:

0x2D	0xE6	0xB8	0x3C	0x11	0x01	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

- 5) Verify CRC32:

CRC32 = 0x2DE6B83C

Calculate CRC32, skip first 4 bytes:

CRC32(0x11, 0x01 ... 0x00) = 2de6b83c

Check if both values are identical. If not, reject the packet.

- 6) Identify *CMD*:

0x2D	0xE6	0xB8	0x3C	0x11	0x01	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

- 7) If *RESP* flag is SET, check value of *RESP_CODE*:

0x2D	0xE6	0xB8	0x3C	0x11	0x01	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00	0x00
------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------

- 8) Get *DATA* (*RESP* flag NOT SET) / *RESP_DATA* (*RESP* flag SET):