Dissecting HomeMatic AES

Due to the discovery of the leaked default HomeMatic AES-key my interest in the AES signing-process was once again renewed, so I decided to take a fresh look at it.

First we take a look at the messages involved in a signed command-execution:

Messages involved

- Initial command-message from the initiator to the device (*m-frame*):

```
0E 03 A0 11 68EA14 1ED017 02 01C80000
len | flg | sender recvr   | \params/
   cnt  type            subtype
```

- Signing request (challenge) from the device to the initiator (*c-frame*):

```
11 03 A0 02 1ED017 68EA14 04 D962D9FB2B03 00
len | flg | sender recvr   | \challenge?/ kno
   cnt  type            sig.req.
```

- Signing response from the initiator to the device (*r-frame*):

```
19 03 A0 03 68EA14 1ED017 344305154D33A8766DBAE938311FA514
len | flg | sender recvr  \---------AES-payload----------/
   cnt  type
```

- ACK-message from the device to the initiator, containig device-authentication (*a-frame*):

```
12 03 80 02 1ED017 68EA14 0101C80016 8B0C277F
len | flg | sender recvr  \ack-data/ \-auth-/
   cnt  type
```

The messages were captured when switching on the output of a HM-LC-Sw1-PI-2 plug with a HM-CFG-USB-2 using the default key.

Dissecting the messages

The obvious approach -- decrypting the AES-payload with the key -- did not result in anything which could be resembled to the input-messages, so I started playing around with the key and the challenge, replacing and XORing them together.

XORing the first 6 bytes of the key with the challenge and then decrypting the payload with the modified key produced the following "plaintext":

`8ac4277f2d41c928a735168946534aa1`

Comparing this to the *a-frame*, we can see that two bytes from the auth-part are identical (`27 7f`):

`12 03 80 02 1ED017 68EA14 0101C80016 8B0C277F`

Repeating this process with different messages confirmed that the last 2 or 3 bytes of the *a-frame* were included in the decrypted plaintext.

Replacing the two "wrong" bytes with the "correct" bytes from the *a-frame* results in this:

`8b0c277f`2d41c928a735168946534aa1

If this is then decrypted again, the following plaintext is produced:

`2c11703e980e`**`03a01168ea141ed01702`**

This contains the first 10 bytes (excluding length) of the *m-frame* in bytes 7 to 16 and what seems to be a timestamp or counter in bytes 1 to 6, so it seems to be decrypted correctly.

But how are the two magic bytes `8b0c` produced? XORing them to the bytes found in their place produces:

`8b0c ^ 8ac4 =` **`01c8`**

These two bytes are the non-null parameters of the *m-frame*:

`0E 03 A0 11 68EA14 1ED017 02` **`01C80000`**

Testing with different messages confirmed that the parameters of the *m-frame* (padded with `0x00` at the end to fill 16 bytes) are XORed to the result of the first decryption before the second decryption, sort of like an IV used in CBC-mode.

HomeMatic AES signature verification algorithm

Based on these findings, the algorithm to verify the signature is:

1. A temporary key is built by XORing the first 6 bytes of the key with the challenge (bytes 12-17 of the *c-frame*).
2. The parameters of the *m-frame* are used as an IV, padded with `0x00` at the end to fill 16 bytes.
3. The AES-payload *P* of the *r-frame* is decrypted with the temporary key, producing *Pd*.
4. The IV is XORed to the decrypted *Pd*, resulting in *Pd^*. The first 4 bytes of *Pd^* will be sent in the ACK-message.
5. The resulting *Pd^* is decrypted (again) with the temporary key, producing *Pd^d*.
6. The end-result *Pd^d* should contain the beginning of the *m-frame* (starting with the counter) which caused the challenge-response-exchange starting at byte 7. Bytes 1 to 6 seem to be the local timestamp or a counter of the device generating the response. The `RPTED`-flag should be masked out in both the original and the decrypted message before comparing, as the sender can't know the value of the flag at the recipient (`flg & 0xbf`)!

Acknowledgments

Thanks to:

- Sathya Laufer from the Homegear project for providing invaluable input and ideas
- The Fhem project which introduced me to HomeMatic and especially Martin Pittner for the hard work of reverse-engineering and implementing the BidCoS-protocol
- The person who posted the default AES-key in November 2014
- eQ-3 for HomeMatic and building a secure authentication mechanism which seems quite attack-proof (if the key is changed by the user)

No HomeMatic-devices were harmed in this process.

2015-06-07, Michael Gernoth < gernoth at cs.fau.de >