Unlocking Keeloq

Rogan Dawes – Researcher
Orange Cyberdefense
Remote controls

Press button of valid transmitter (if menu locked)

11.1. Add remotes
11.2. Delete remotes...... 11.2.1. Delete remote by ID
11.2.2. Delete remote button
11.2.3. Delete remote by button
11.2.4. Delete not present
11.2.5. Delete all remotes

11.3. Edit remote button
11.4. Autolearn
11.5. Lock Tx menu
11.6. Onboard receiver enable/disable
Keeloq

- Secure remote control systems can only be implemented if two conditions are met.
  - A large number of possible combinations must be available
  - The system may never respond twice to the same transmitted code
Over the air (ASK/OOK)

50% Duty Cycle
Preamble $T_P$

Header $T_H$

Encrypted Portion of Transmission $T_{HOP}$

Fixed Portion of Transmission $T_{FIX}$

Guard Time $T_G$

34 bits of Fixed Portion

<table>
<thead>
<tr>
<th>Repeat (1 bit)</th>
<th>VLow (1 bit)</th>
<th>Button Status</th>
<th>Serial Number (28 bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>S2 S1 S0 S3</td>
<td></td>
</tr>
</tbody>
</table>

32 bits of Encrypted Portion

<table>
<thead>
<tr>
<th>Button Status</th>
<th>OVR (2 bits)</th>
<th>DISC (10 bits)</th>
<th>Sync Counter (16 bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2 S1 S0 S3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

66 Data bits Transmitted MSb first.
Key Derivation Function (KDF) – Normal Learn

Manufacturer knows
- KDF
- Manufacturer key

Transmitter contains
- Shared key

Sends with each transmission
- Transmitter serial number
- Encrypted counter

Receiver contains
- KDF
- Manufacturer key

Receives
- Transmitter serial number

Derives
- Shared key

Checks
- Counter
uint64_t normal_keygen(uint32_t serial) {
    static uint64_t key;
    static uint32_t cached = 0;

    // make sure the function code is masked out
    serial &= 0xffffffff;

    if (serial == cached)
        return key;

    key = keeloq_decrypt(serial | 0x60000000, mkey_);
    key = key << 32 | keeloq_decrypt(serial | 0x20000000, mkey_);

    cached = serial;

    return key;
}
What attacks have been tried?

- **Cryptanalysis**
  - Specific weaknesses due to implementation flaws

- **Side Channel**
  - Recover key material from transmitter or receiver through power analysis

- **Replay**
  - Jam one transmission while recording it
  - Jam (and record) a second transmission while replaying the first
Black Magic Probe V2.1
Open Source JTAG & SWD GNU Debugger and Programmer with built in GDB server & UART
SVD Loader

SVD-Loader for Ghidra automates the entire generation of peripheral structs and memory maps for over 650 different microcontrollers: By parsing so-called SVD files (CMSIS System View Description) SVD-Loader is able to automatically annotate all peripherals of the controller, simplifying reverse-engineering of ARM firmwares significantly.

```c
bVar1 = (bool)isCurrentModePrivileged();
if (bVar1) {
    enableIRQinterrupts((iVar3 & 1) == 1);
}
do {
    while (((DAT_500000014 & 0x200) == 0);
   _DAT_40000000 = DAT_0000dd74 & (_DAT_40000000 | 0xc0000000);
   _DAT_40000020 = DAT_0000dd78 & _DAT_40000200 | DAT_00000000;
   _DAT_50000008 = (_DAT_50000000 & 0xffcc | 0xc9) & 0xffbf;
   _DAT_50000010 = _DAT_50000010 & 0xffff;
   return;
   bVar1 = (bool)isCurrentModePrivileged();
   if (bVar1) {
       enableIRQinterrupts((iVar4 & 1) == 1);
   }
   do {
       while (((iVar2 & 0x200) == 0);
       iVar2 = read_volatile2(CRG_TOP.SYS_STAT_REG);
       while (((iVar2 & 0x200) == 0);
       iVar4 = read_volatile4(BLE.BLE_RMBLECNTL_REG);
       write_volatile4(BLE.BLE_RMBLECNTL_REG, DAT_00000000 & iVar4);
       iVar4 = read_volatile4(BLE.BLE_CNTL2_REG);
       write_volatile4(BLE.BLE_CNTL2_REG, DAT_00000000 & iVar4 | DAT_00000000);
```
<table>
<thead>
<tr>
<th>Location</th>
<th>Label</th>
<th>Code Unit</th>
<th>String View</th>
<th>Stri...</th>
<th>Le...</th>
<th>Is Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>0004343</td>
<td>ds</td>
<td>&quot; events.txt&quot;</td>
<td>&quot; events.txt&quot;</td>
<td>string</td>
<td>12</td>
<td>true</td>
</tr>
<tr>
<td>000435b</td>
<td>ds</td>
<td>&quot; Sherlotronics PTV/L...</td>
<td>&quot; Sherlotronics PTV/LTD Even...</td>
<td>string</td>
<td>58</td>
<td>true</td>
</tr>
<tr>
<td>0004505</td>
<td>ldr</td>
<td>r0, [r0, #0x0]</td>
<td>&quot;h' Ox&quot;</td>
<td>string</td>
<td>6</td>
<td>false</td>
</tr>
<tr>
<td>00045fb</td>
<td>ldr</td>
<td>r0, [r0, #0x0]</td>
<td>&quot;h' 0x&quot;</td>
<td>string</td>
<td>6</td>
<td>false</td>
</tr>
<tr>
<td>00047f4</td>
<td>s_Relay2_...</td>
<td>ds</td>
<td>&quot;Relay2&quot;</td>
<td>&quot;Relay2&quot;</td>
<td>string</td>
<td>7</td>
</tr>
<tr>
<td>00047ff</td>
<td>ds</td>
<td>&quot; Date: %s Time: % U...</td>
<td>&quot; Date: %s Time: % s Unit:%s...</td>
<td>string</td>
<td>46</td>
<td>false</td>
</tr>
<tr>
<td>0004830</td>
<td>s_Relay1_...</td>
<td>ds</td>
<td>&quot;Relay1&quot;</td>
<td>&quot;Relay1&quot;</td>
<td>string</td>
<td>7</td>
</tr>
<tr>
<td>0004867</td>
<td>PTR_GPIO...</td>
<td>addr Peripherals::GPIOB</td>
<td>@RELAY:</td>
<td>string</td>
<td>8</td>
<td>true</td>
</tr>
<tr>
<td>0004900</td>
<td>LAB_000D...</td>
<td>ldrb r0, [r5, #0x0] = LAB_2...</td>
<td>&quot;(x8bx&quot;</td>
<td>string</td>
<td>7</td>
<td>false</td>
</tr>
<tr>
<td>0004939</td>
<td>ldr</td>
<td>r1, [r0, #0x0] = DAT_20...</td>
<td>&quot;hi' Aha'&quot;</td>
<td>string</td>
<td>8</td>
<td>false</td>
</tr>
<tr>
<td>0004c93</td>
<td>ds</td>
<td>&quot; Again&quot;</td>
<td>&quot; Again&quot;</td>
<td>string</td>
<td>7</td>
<td>false</td>
</tr>
<tr>
<td>0004c9c</td>
<td>s_Press_0...</td>
<td>ds</td>
<td>&quot;Press&quot;</td>
<td>&quot;Press&quot;</td>
<td>string</td>
<td>6</td>
</tr>
<tr>
<td>0004cbb</td>
<td>ds</td>
<td>&quot; Code is &quot;</td>
<td>&quot; Code is &quot;</td>
<td>string</td>
<td>10</td>
<td>true</td>
</tr>
<tr>
<td>0004cc8</td>
<td>s_in_use_...</td>
<td>ds</td>
<td>&quot;in use &quot;</td>
<td>&quot;in use &quot;</td>
<td>string</td>
<td>9</td>
</tr>
<tr>
<td>0004cd7</td>
<td>PTR_PTR...</td>
<td>addr PTR_DAT_20000018</td>
<td>&quot; RELAY1&quot;</td>
<td>string</td>
<td>8</td>
<td>false</td>
</tr>
<tr>
<td>00050e8</td>
<td>LAB_000D...</td>
<td>cmp r4, r8</td>
<td>&quot;DELETING&quot;</td>
<td>string</td>
<td>9</td>
<td>false</td>
</tr>
<tr>
<td>00050f4</td>
<td>s_REMOTE...</td>
<td>ds</td>
<td>&quot;REMOTE &quot;</td>
<td>&quot;REMOTE &quot;</td>
<td>string</td>
<td>9</td>
</tr>
<tr>
<td>0005100</td>
<td>s_DELETE...</td>
<td>ds</td>
<td>&quot;DELETED&quot;</td>
<td>&quot;DELETED&quot;</td>
<td>string</td>
<td>9</td>
</tr>
<tr>
<td>0005108</td>
<td>s_REMOTE...</td>
<td>ds</td>
<td>&quot;REMOTE&quot;</td>
<td>&quot;REMOTE&quot;</td>
<td>string</td>
<td>7</td>
</tr>
<tr>
<td>0005113</td>
<td>DAT_000...</td>
<td>undefined4 200001ACH</td>
<td>&quot; RELAY1&quot;</td>
<td>string</td>
<td>9</td>
<td>false</td>
</tr>
<tr>
<td>000511c</td>
<td>s_RELAY2...</td>
<td>ds</td>
<td>&quot;RELAY2&quot;</td>
<td>&quot;RELAY2&quot;</td>
<td>string</td>
<td>8</td>
</tr>
<tr>
<td>0005133</td>
<td>PTR_GPIO...</td>
<td>addr Peripherals::GPIOB</td>
<td>@DELETE</td>
<td>string</td>
<td>8</td>
<td>true</td>
</tr>
</tbody>
</table>

Filter:  

- Auto Label
- Offset: 0 Dec
- Preview: "Press"

Include Alignment Nulls

Truncate if Needed

Make String
Make Char Array
Tips for reverse engineering crypto code – David Lodge

https://www.pentestpartners.com/security-blog/reverse-engineering-keys-from-firmware-a-how-to/

TL;DR

It is possible to reverse engineer keys from firmware with some tips:

1. Always looks for strings/constants.
2. Make guesses about the original source.
3. Find a function you can recognise and work backwards to identify other functions.
4. It helps if they use open-source code so you can crib from it.
Decoding the Keeloq code word
ESPHome is a system to control your ESP8266/ESP32 by simple yet powerful configuration files and control them remotely through Home Automation systems.

```yaml
sensor:
  - platform: dht
    pin: D2
    temperature:
      name: "Temperature"
    humidity:
      name: "Humidity"
```

**Living Room**

- Temperature: 15.6 °C
- Humidity: 63%
Porting to STM32

$ git diff --stat=120 84b40f90..stm32
esphome/boards.py
esphome/components/logger/__init__.py
esphome/components/logger/logger.cpp
esphome/components/logger/logger.h
esphome/components/remote_receiver/remote_receiver.h
esphome/components/remote_receiver/remote_receiver_es8266.cpp
esphome/components/uart/uart.cpp
esphome/components/uart/uart.h
esphome/components/uart/uart_stm32.cpp
esphome/const.py
esphome/core/application.cpp
esphome/core/application_stm32.cpp
esphome/core/config.py
esphome/core/esp32.cpp
esphome/core/helpers.cpp
esphome/core/helpers.h
esphome/core/preferences.cpp
esphome/core/preferences.h
esphome/core/stm32.cpp
esphome/pins.py
platformio.ini
21 files changed, 594 insertions(+), 22 deletions(-)
Implementing Keeloq and Normal KDF

```bash
$ git diff --stat=120 keelog^

esphome/components/hcs301/__init__.py | 33 +++++++
esphome/components/hcs301/hcs301.cpp  | 186 ++++++++++++++++++++++++++++++++++++++++++++++++++++++
esphome/components/hcs301/hcs301.h    | 50 ++++++++   
esphome/components/keeloq_normal_crypter/__init__.py| 22 +++++++
esphome/components/keeloq_normal_crypter/keeloq_normal_crypter.cpp | 104 +++++++++++++++++++++++
esphome/components/keeloq_normal_crypter/keeloq_normal_crypter.h  | 30 ++++++++   
esphome/components/remote_base/__init__.py      | 54 +++++++++++
esphome/components/remote_base/keeloq_protocol.cpp | 110 +++++++++++++++++++++++
esphome/components/remote_base/keeloq_protocol.h    | 51 ++++++++++++   
9 files changed, 640 insertions(+)
```
Demonstration of ESPHome/Keeloq

```
$ bt /dev/ttyACM2
```
Bringing it all online
ESPHome Client API

$ git diff --stat 5cb56bc6..api_client
esphome/components/api/__init__.py                       7 +-
esphome/components/api/api_connection.cpp               166 +++++++++---
esphome/components/api/api_connection.h                76 +-----
esphome/components/api/api_pb2.cpp                      660 +--------------------------------------
esphome/components/api/api_pb2.h                        2 +-
esphome/components/api/api_pb2_service.cpp              24 +-
esphome/components/api/api_pb2_service.h                2 +-
esphome/components/api/api_server.cpp                   54 +--
esphome/components/api/api_server.h                     15 +-
esphome/components/api/api_client/__init__.py           73 +-----
esphome/components/api/api_client/api_client_connection.cpp 436 +------------------------------------------------
esphome/components/api/api_client/api_client_connection.h 164 +----------
esphome/components/api/api_client/api_pb2_client.cpp    545 +--------------------------------------
esphome/components/api/api_client/api_pb2_client.h       139 +-------
esphome/components/api/api_client/binary_sensor.py      29 +
esphome/components/api/api_client/proto_client.h        29 +
esphome/components/api/api_client/sensor.py             31 +
esphome/components/api/api_client/switch/__init__.py     29 +
esphome/components/api/api_client/switch/api_switch.cpp  27 +
esphome/components/api/api_client/switch/api_switch.h    25 ++
esphome/components/api/api_client/text_sensor.py         27 ++
script/api_protobuf/api_protobuf.py                     120 +-------
22 files changed, 2233 insertions(+), 447 deletions(-)
Home Assistant and Keeloq Remotes
remote_receiver:
  id: receiver
  pin:
    number: PA 3
    mode: INPUT
  buffer_size: 200
  tolerance: 30%
  on_keeloq:
    then:
      - lambda: |
          char buff[20];
          if (id(keeloq_crypter).decrypt(x)) {
            snprintf(buff, sizeof(buff), "%07x:%1x:%04X:%c:%c",
                       x.serial, x.button, x.sync, x.low ? 'L' : 'N', x.repeat ? 'R' : 'F');
          } else {
            snprintf(buff, sizeof(buff), "%07x:%1x:%c:%c",
                       x.serial, x.button, x.low ? 'L' : 'N', x.repeat ? 'R' : 'F');
          }
          string buffAsStdStr = buff;
          id(keeloq_remote).publish_state(buff);
hcs301:
  id: hcs301_id
  power_pin: PB_15
  clock_pin: PB_14
  pwm_pin: PB_13

script:
  - id: program_hcs301
    mode: single
    then:
      - lambda: |
        uint64_t hcs301_key = id(keeloq_crypter).normal_keygen(0x0DA342B);
        if (id(hcs301_id).program(0x0DA342B, 0x0, hcs301_key)) {
          ESP_LOGD("hcs301", "Successfully programmed");
        }
Outstanding features

- Persistent recording of counters and replay detection
- Desynchronisation recovery in the client API
- Implementation of other entity types in client API
- HCS301 initial sequence no
Code

- ESPHome
- [https://github.com/rogandawes/esphome](https://github.com/rogandawes/esphome)
- Branches stm32, keeloq and api_client
Thank You!

Rogan Dawes
@RoganDawes
rogan.dawes@orangecyberdefense.com
Questions?