One Car, Two Frames:
Attacks on Hitag–2 Remote Keyless Entry Systems Revisited

Ryad BENADJILA$^1$
José LOPES-ESTEVES$^2$
Mathieu RENARD$^2$
Chaouki KASMI$^2$

$^1$Thales C&S, ryadbenadjila@gmail.com
$^2$ANSSI, forename.name@ssi.gouv.fr

August 15, 2017
Vancouver, BC, Canada
Introduction
Car locking systems

- Used to open/close a car and for anti-theft immobilizers.
Car locking systems

- Used to **open/close** a car and for **anti-theft immobilizers**.
Car locking systems

- Used to open/close a car and for anti-theft immobilizers.
Car locking systems

- **This talk**: focus on open/close Remote Keyless Entry systems.
Context
Remote Keyless Entry

- **RKE:**
  1. Monodirectional communication between remote key and ECU.
Remote Keyless Entry

- **RKE:**
  1. **Monodirectional communication** between remote key and ECU.
  2. Threats: recording,
Remote Keyless Entry

**RKE:**

1. **Monodirectional communication** between remote key and ECU.
2. Threats: recording, replaying,
Remote Keyless Entry

- **RKE:**
  1. Monodirectional communication between remote key and ECU.
  2. Threats: recording, replaying, jamming,
Remote Keyless Entry

- **RKE:**
  1. Monodirectional communication between remote key and ECU.
  2. Threats: recording, replaying, jamming, spoofing.
Remote Keyless Entry

<table>
<thead>
<tr>
<th>UID</th>
<th>Key</th>
<th>Counter</th>
<th>Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID1</td>
<td>K1</td>
<td>10</td>
<td>0,1,5,6</td>
</tr>
<tr>
<td>UID2</td>
<td>K2</td>
<td>200</td>
<td>0,1,2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

ECU

KEY 1

KEY 2

UID1, CNTR1, BTN1, AUTH1

UID2, CNTR2, BTN2, AUTH2

ECU

AUTH1

K1

AUTH2

K2

UID1 CNTR1 BTN1

UID2 CNTR2 BTN2
Remote Keyless Entry

| Introduction | Context | Radio signal analysis | Hitag-2 | New attacks | Conclusion |

Remote Keyless Entry

One Car, Two Frames - WOOT’17 (August 15, 2017 - Vancouver, BC, Canada)
Remote Keyless Entry

<table>
<thead>
<tr>
<th>UID</th>
<th>Key</th>
<th>Counter</th>
<th>Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID1</td>
<td>K1</td>
<td>11</td>
<td>0,1,5,6</td>
</tr>
<tr>
<td>UID2</td>
<td>K2</td>
<td>200</td>
<td>0,1,2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

BTN1 ∈ {0,1,5,6}?
CNTR1 ∈ [10, 10 + δ]?

ECU

KEY 1

KEY 2

AUTH_ECU =?= AUTH1
USENIX 2016: attacks on RKE systems

- **USENIX 2016 article**: “Lock It and Still Lose It - On the (In)Security of Automotive Remote Keyless Entry Systems”.

- Two attacks are discussed:
  1. **Volkswagen** – good crypto but master keys are shared amongst all vehicles since 2000!
  2. **PCF7946** – Philips/NXP transponder using **Hitag-2**. Correlation attack unveiled.
**USENIX 2016: attacks on RKE systems**

- **Goal:** setup the attack targeting the **PCF7946**.
  1. Capture and **decode** the radio frames.
  2. Implement the **correlation attack**.
  3. Find the **secret key** using the attack.
  4. Craft **valid radio frames** and profit.

- **Constraints:** **black-box** approach.
  - Breaking the car was not an option!
  - Neither **invasive nor semi-invasive attacks** on the **PCF7946** considered.
    - Time and resource **costly**!
Radio signal analysis
From RF signal to bits

- Useful information to be gathered:
  - Central frequency and channel bandwidth.
  - Modulation.
  - Channel encoding.
  - Packet format.

- White-box analysis:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Working frequency</td>
<td>ISM 433 MHz</td>
</tr>
<tr>
<td>Modulation</td>
<td>ASK/FSK</td>
</tr>
<tr>
<td>Channel encoding</td>
<td>Manchester/NRZ</td>
</tr>
<tr>
<td>Packet format</td>
<td>see USENIX 2016</td>
</tr>
</tbody>
</table>
Demodulation: spectral analysis

- Modulation: ASK (Amplitude Shift Keying).
Decoding
Decoding

![Waveform](image_url)
Decoding
Decoding

Symbol period
Decoding

Results:
- Modulation: ASK.
- Channel encoding: Manchester.
- Observing invariants to get back to the data.
- Using the checksum for sanity check.
Hitag-2
The Hitag-2 algorithm

- Late 90’s stream cipher from Philips (NXP).
- Hardware reverse engineered in 2007.
The Hitag-2 algorithm

- Late 90’s stream cipher from Philips (NXP).
- Hardware reverse engineered in 2007.
- Using the algorithm in a RKE context:
The Hitag-2 algorithm

- Late 90’s stream cipher from Philips (NXP).
- Hardware reverse engineered in 2007.
- Using the algorithm in a RKE context:
**Hitag-2**: initialization phase

\[ uid = uid_0 \ldots uid_{31} \text{ (32 bits)} \]

\[ k_{\text{low}} = k_0 \ldots k_{16} \text{ (16 bits)} \]

Internal state Hitag-2 (48 bits)
Hitag-2: initialization phase

$uid = uid_0 \ldots uid_{31}$ (32 bits)

$k_{\text{low}} = k_0 \ldots k_{16}$ (16 bits)

Internal state Hitag-2 (48 bits)
Hitag-2: randomization phase

\[ iv = iv_0 \ldots iv_{31} \text{ (32 bits)} \]

\[ k_{\text{high}} = k_{16} \ldots k_{47} \text{ (32 bits)} \]
Hitag-2: randomization phase

\[ iv = iv_0 \ldots iv_{31} \text{ (32 bits)} \]

\[ k_{\text{high}} = k_{16} \ldots k_{47} \text{ (32 bits)} \]
**Hitag-2: randomization phase**

\[ iv = iv_0 \ldots iv_{31} \text{ (32 bits)} \]

\[ k_{\text{high}} = k_{16} \ldots k_{47} \text{ (32 bits)} \]
Hitag-2: randomization phase

\[ iv = iv_0 \ldots iv_{31} \text{ (32 bits)} \]

\[ k_{\text{high}} = k_{16} \ldots k_{47} \text{ (32 bits)} \]
Hitag-2: randomization phase

\[ iv = iv_0 \ldots iv_{31} \text{ (32 bits)} \]

\[ k_{\text{high}} = k_{16} \ldots k_{47} \text{ (32 bits)} \]
**Hitag-2: randomization phase**

\[ iv = iv_0 \ldots iv_{31} \text{ (32 bits)} \]

\[ k_{\text{high}} = k_16 \ldots k_{47} \text{ (32 bits)} \]
Hitag-2: randomization phase

\[ iv = iv_0 \ldots iv_{31} \text{ (32 bits)} \]

\[ k_{\text{high}} = k_{16} \ldots k_{47} \text{ (32 bits)} \]
Hitag-2: randomization phase

\[ iv = iv_0 \ldots iv_{31} \text{ (32 bits)} \]

\[ k_{\text{high}} = k_{16} \ldots k_{47} \text{ (32 bits)} \]
Hitag-2: nominal phase

\[ f_a(i) = (0xA63C)_i \]

\[ f_h(i) = (0xA770)_i \]

\[ f_s(i) = (0xD949CB0)_i \]

\[ \text{keystream} \]
Hitag-2: nominal phase

\[ f_a(i) = (0xA63C) \]
\[ f_b(i) = (0xA770) \]
\[ f_c(i) = (0xD949CBB0) \]

ks0, ks1

keystream
Hitag-2: nominal phase

\[ f_a(i) = (0xA63C)_i \]
\[ f_b(i) = (0xA770)_i \]
\[ f_b(i) = (0xA770)_i \]
\[ f_b(i) = (0xA770)_i \]
\[ f_a(i) = (0xA63C)_i \]
\[ f_c(i) = (0xD949CB0)_i \]

keystream

\[ L(state) \]
**Hitag-2: nominal phase**

\[ f_a(i) = (0xA63C)_i \]
\[ f_b(i) = (0xA770)_i \]
\[ f_b(i) = (0xA770)_i \]
\[ f_b(i) = (0xA770)_i \]
\[ f_a(i) = (0xA63C)_i \]
\[ f_a(i) = (0xA63C)_i \]
\[ f_c(i) = (0xD94CBB0)_i \]

\[ L(state) \]

 keystream
Hitag-2: the correlation attack

- Introduced by the USENIX 2016 article:
  - Key recovery with 4 to 8 radio frames.
  - The key search space is significantly reduced.
  - Uses key candidates scoring deduced from the observed keystream.
Hitag-2: the correlation attack

- Introduced by the USENIX 2016 article:
  - Key recovery with 4 to 8 radio frames.
  - The key search space is significantly reduced.
  - Uses key candidates scoring deduced from the observed keystream.

- Solving the unknown CNTRH issue:
  - Supposed to be set to zero at manufacturing time.
  - Authors suggest to estimate the vehicle’s age.
Implementing the correlation based cryptanalysis

- Tests on emulated radio frames.
  - Our implementation works.
  - The key is found in a few minutes.
Implementing the correlation based cryptanalysis

- Tests on **emulated radio frames**.
  - Our implementation *works*.
  - The key is found in *a few minutes*.

- Tests on **real radio frames (with unknown CNTRH)**.
  - Cryptanalysis does not converge towards a proper key.
Implementing the correlation based cryptanalysis

- Tests on emulated radio frames.
  - Our implementation works.
  - The key is found in a few minutes.

- Tests on real radio frames (with unknown CNTRH).
  - Cryptanalysis does not converge towards a proper key.

- Our Hitag-2 RKE system might be different!
  - We need to understand the discrepancies.
New attacks
Black box reverse engineering

How can it be performed?

- We had access to the vehicle but no access to the ECU.
- No NDA with NXP: neither datasheets nor SDKs.
Black box reverse engineering

- How can it be performed?
  - We had access to the vehicle but no access to the ECU.
  - No NDA with NXP: neither datasheets nor SDKs.

- We found programmable blank keys containing the PCF7946!
  - They use the manufacturing default key \texttt{0x4f4e4d494b52}.
Finding the $iv$ format: a black box approach

- Brute forcing the $2^{32}$ $iv$ and finding explicit patterns for observed $ks$, with fixed and known $id$ and $k$. 

![Diagram](https://via.placeholder.com/150)
Finding the $iv$ format: the discrepancies
Finding the \textit{iv} format: the discrepancies

- Explains why the USENIX 2016 correlation attack fails.
Uncovering an ECU mitigation

<table>
<thead>
<tr>
<th>UID</th>
<th>Key</th>
<th>Counter</th>
<th>Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID1</td>
<td>K1</td>
<td>10</td>
<td>0,1,5,6</td>
</tr>
<tr>
<td>UID2</td>
<td>K2</td>
<td>200</td>
<td>0,1,2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

UID1, CNTRL1 = 9, BTN1, AUTH1
Uncovering an ECU mitigation

<table>
<thead>
<tr>
<th>UID</th>
<th>Key</th>
<th>Counter</th>
<th>Button</th>
</tr>
</thead>
<tbody>
<tr>
<td>UID1</td>
<td>K1</td>
<td>10</td>
<td>0,1,5,6</td>
</tr>
<tr>
<td>UID2</td>
<td>K2</td>
<td>200</td>
<td>0,1,2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

CNTR1 = 9 < 10

ECU resynchronization
Uncovering an ECU mitigation

- Resynchronization with near-field 125 KHz when starting the engine.
Optimized exhaustive search

- Uses two triplets \((id, iv, ks)\):
  - Searches over \(2^{48}\) keys the one realizing the observed keystreams.
  - Implementation of a heavily parallelized and optimized brute-forcer on CPU and GPU (in OpenCL).
Optimized exhaustive search

- Uses two triplets \((id, iv, ks)\):
  - Searches over \(2^{48}\) keys the one realizing the observed keystreams.
  - Implementation of a heavily parallelized and optimized brute-forcer on CPU and GPU (in OpenCL).

- Tested on Amazon EC2 instances:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeForce GTX 780Ti</td>
<td>18 hours</td>
</tr>
<tr>
<td>One Amazon EC2† instance</td>
<td>45 minutes</td>
</tr>
<tr>
<td>Three Amazon EC2† instances</td>
<td>15 minutes</td>
</tr>
</tbody>
</table>

†p2.16xlarge: 16 Tesla K80, 128 CPU
Optimized exhaustive search

- Uses **two triplets** \((id, iv, ks)\):
  - Searches over \(2^{48}\) keys the one realizing the observed keystreams.
  - Implementation of a heavily parallelized and optimized brute-forcer on CPU and GPU (in OpenCL).

- Tested on **Amazon EC2** instances:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>GeForce GTX 780Ti</td>
<td>18 hours</td>
</tr>
<tr>
<td>One Amazon EC2(^\d) instance</td>
<td>45 minutes</td>
</tr>
<tr>
<td>Three Amazon EC2(^\d) instances</td>
<td>15 minutes</td>
</tr>
</tbody>
</table>

\(^\d\)p2.16xlarge: 16 Tesla K80, 128 CPU

- How to deal with the **unknown part of CNTRH**?
Hitag-2 equivalent keys

- **Masking can be inserted during the randomization phase.**

\[ \hat{iv} = iv \oplus M \]

\[ \hat{k}_{\text{high}} = k_{\text{high}} \oplus M \]

\[ f_a(i) = (0xA63C), \]

\[ f_6(i) = (0xA770), \]

\[ f_6(i) = (0xA770), \]

\[ f_6(i) = (0xA770), \]

\[ f_6(i) = (0xA770), \]

\[ f_a(i) = (0xA63C), \]

\[ f_6(i) = (0xD949CBB0), \]

\[ \text{Shift Register} \]
Hitag-2 equivalent keys

Many equivalent keys generating the same keystream can be exposed through *iv* masking.
Many equivalent keys generating the same keystream can be exposed through *iv* masking.

\[ \hat{k}_1 = \hat{k}_2 = \hat{k}_3 \]

\[ k_{0...15} \oplus k_{16...33} \oplus M_i \]

\[ k_{34...47} \]

\[ id = \text{UID} \]

\[ k_s = \text{KS} \]
**Hitag-2 equivalent keys**

- Many equivalent keys generating the same keystream can be exposed through $iv$ masking.

- Particular case of interest: when the mask is $CNTRH$. 

![Diagram](diagram.png)
Hitag-2 equivalent keys

- Many equivalent keys generating the same keystream can be exposed through $iv$ masking.

- An exhaustive search with equivalent $\hat{iv}$ produces an equivalent key $\hat{k}$ masked with CNTRH.

\[
\hat{k} = k \oplus (0^{16} \parallel CNTRH \parallel 0^{14})
\]
Hitag-2 equivalent keys

- Many equivalent keys generating the same keystream can be exposed through $iv$ masking.

- An exhaustive search with equivalent $\hat{iv}$ produces an equivalent key $\hat{k}$ masked with CNTRH.

- No need to find the real key $k$ to craft legitimate frames!
New attacks 1/2: capture two frames and guess

- **Without** ECU resynchronization.
New attacks 1/2: capture two frames and guess

- **Without** ECU resynchronization.

\[ \hat{k} = \text{Exhaustive search of } 2^{48} \]
\[ \Rightarrow 15 \text{ minutes on Amazon EC2} \]
New attacks 1/2: capture two frames and guess

- **Without** ECU resynchronization.
New attacks 2/2: recapture and adapt

- With ECU resynchronization.

\[
\hat{k} = \text{Exhaustive search of } 2^{18} \implies \text{15 minutes on a laptop}
\]

\[
\hat{k}^\prime \leq (1024 - \text{CNTRL}) \text{ frames OK} > (1024 - \text{CNTRL}) \implies \text{increment}
\]
New attacks 2/2: recapture and adapt

- **With** ECU resynchronization.

\[ \hat{k'} = \text{Exhaustive search of } 2^{18} \]
\[ \Rightarrow 15 \text{ minutes on a laptop} \]
New attacks 2/2: recapture and adapt

- **With ECU resynchronization.**

\[
\hat{k}' \leq (1024 - \text{CNTRL}) \text{ frames OK}
\]

\[
> (1024 - \text{CNTRL}) \Rightarrow \text{increment}
\]
Conclusion
Conclusion

Results:

- A hardened RKE Hitag-2 exposed.
  - Mitigation through ECU resynchronization.
Conclusion

Results:

- A hardened RKE Hitag-2 exposed.
  - Mitigation through ECU resynchronization.

- Attack cost \( \approx 20 + 45 + 100 \) \( \text{€} \).

- Attack complexity = 2 RF frames, +1 with the ECU mitigation.
Conclusion

Results:

- A hardened RKE Hitag-2 exposed.
  - Mitigation through ECU resynchronization.

- Attack cost \(\approx 20 + 45 + 100 \text{ €} \).

- Attack complexity = 2 RF frames, +1 with the ECU mitigation.

Obsolete and proprietary cryptography is broken:
- Time to make a change!
One Car, Two Frames: Attacks on Hitag-2 Remote Keyless Entry Systems Revisited