Overview of some automotive RKE systems

Pierre Pavlidès

OWASP Gothenburg Day 2016

November 24, 2016
Before we start

Slides at http://r.rogdham.net/26

Pierre Pavlidès

@rogdham

Lock It and Still Lose It—On the (In)Security of Automotive Remote Keyless Entry Systems

Flavio D. Garcia and David Oswald, University of Birmingham; Timo Kasper, Kasper & Oswald GmbH; Pierre Pavlidès, University of Birmingham

https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/garcia
Before we start
1. RKE from scratch
2. The VW systems
3. The Hitag2 system
Agenda

1. RKE from scratch
2. The VW systems
3. The Hitag2 system
Car security mechanisms

Going inside the car:
→ Mechanical lock
→ RKE system
→ Smart key

Starting the engine and drive away:
→ Be inside the car first
→ Immobiliser (transponder)
→ Mechanical lock / ignition button

Once you are in the car, game over
→ Attacks on transponder
→ Access the onboard computer

In this talk: attacks on RKE systems

NOT in this talk: newest car “features” such as access over Internet
Remote overview

Overview of some automotive RKE systems
Remote overview
RKE overview

RKE systems use a very simple protocol:

- Owner pushes a button
- The remote sends a message to the car
- The car reacts accordingly

Only a single one-way message

→ Usual security protocols (e.g. challenge/response) don’t work

Since it seems simple, let’s create an RKE protocol from scratch
Action: open/close the doors, etc.

Tell the car if we want to close or open the doors:

< btn >

Issue

In a parking lot, we will open all cars at once!
Car or remote identifier

Add the car or remote identifier:

\(<UID|btn>\)

Using the remote identifier allows to invalidate specific remotes

---

**Attack scenario**

1. Eavesdrop a close signal
2. Create an open signal
Generate random shared key $key_{UID}$ when the remote is linked with the car

\[
\left< UID \left| \begin{array}{c}
\text{btn} \\
\end{array} \right| \text{MAC}_{key_{UID}}(M) \right>
\]

Does not matter if $M$ includes the $UID$

---

**Attack scenario**

Replay attacks
## Freshness

<table>
<thead>
<tr>
<th>Token</th>
<th>Issue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random(^1)</td>
<td>Car needs to remember all used tokens</td>
</tr>
<tr>
<td>Time-based(^2)</td>
<td>Precise clock in remote (and car)</td>
</tr>
<tr>
<td>Counter</td>
<td>Overflow, desynchronisation</td>
</tr>
</tbody>
</table>

Use a counter with a validity window

\[
\langle UID || btn, ctr || MAC_{key_{UID}}(M) \rangle
\]


A rolling code [...] is used in keyless entry systems to prevent replay attacks

\(^1\) e.g. anti-CSRF (ESAPI...)

\(^2\) e.g. Google Authenticator (RFC 6238)
Any remaining problems?

\[ < UID || \{ M \} || \text{btn, ctr} || \text{MAC}_{\text{key}_{UID}}(M) > \]

Confidentiality
→ Not sure we need it

Integrity: MAC

Availability
→ Repeat the rolling code several times, hope for the best
→ Worst case scenario: owner uses mechanical lock
Agenda

1. RKE from scratch
2. The VW systems
3. The Hitag2 system
The VW systems

- Amarok, (New) Beetle, Bora, Caddy, Crafter, e-Up, Eos, Fox, Golf 4, Golf 5, Golf 6, Golf Plus, Jetta, Lupo, Passat, Polo, T4, T5, Scirocco, Sharan, Tiguan, Touran, Up

- AA1, Q3, R8, S3, TT, others

- Alhambra, Altea, Arosa, Cordoba, Ibiza, Leon, MII, Toledo

- ACity Go, Roomster, Fabia 1, Fabia 2, Octavia, Superb, Yeti
Dump ECU firmware and start reverse engineer assembly

```assembly
; SUBROUTINE -------------------------------

sub_F5C4:
  pshd
  pshx
  leas $C,sp
  anda $3F ; ?
  clr
  addd $8000
  bcc loc_F5D2
  inx

loc_F5D2:
  std 4,sp
  ldd $14,sp
  ldx $12,sp
  subd $E,sp
  sbex $C,sp
```

Pierre Pavlidès

Overview of some automotive RKE systems

17 / 42
VW-1 scheme

Used approx. until 2005

\[ < f(UID) \mid g(ctr) \mid btn > \]

\( f \) and \( g \) are deterministic functions

**Issues**

No shared secret, no integrity checks...

\[ \rightarrow \text{Security through obscurity (obfuscation)} \]
VW-2 scheme

Used approx. since 2004 (VW-2)

\[
< start_2 || AUT64_{key_2} (UID, ctr, btn) || btn >
\]

*start*$_2$ is a fixed prefix
AUT64 is a cipher

**Issue**

Use encryption for integrity

**Issue**

One worldwide unique key
VW-2, VW-3, VW-4 schemes

Used approx. since 2004 (VW-2) / 2006 (VW-3) / 2009 (VW-4)

\[
\begin{align*}
\text{VW-2:} & \quad < \text{start}_2||\text{AUT64}_{\text{key}_2}(\text{UID}, \text{ctr}, \text{btn})||\text{btn} > \\
\text{VW-3:} & \quad < \text{start}_3||\text{AUT64}_{\text{key}_3}(\text{UID}, \text{ctr}, \text{btn})||\text{btn} > \\
\text{VW-4:} & \quad < \text{start}_4||\text{XTEA}_{\text{key}_4}(\text{UID}, \text{ctr}, \text{btn})||\text{btn} >
\end{align*}
\]

Issue

Still worldwide unique key
The VW systems

\[
\begin{align*}
\text{VW-1:} & \quad \langle f(UID)\|g(ctr)\|btn \rangle \\
\text{VW-2:} & \quad \langle \text{start}_2\|\text{AUT64}_{key_2}(UID, ctr, btn)\|btn \rangle \\
\text{VW-3:} & \quad \langle \text{start}_3\|\text{AUT64}_{key_3}(UID, ctr, btn)\|btn \rangle \\
\text{VW-4:} & \quad \langle \text{start}_4\|\text{XTEA}_{key_4}(UID, ctr, btn)\|btn \rangle 
\end{align*}
\]

**Attack scenario**

Possible to clone a remote if we capture a single rolling code

**Impact**

Most VW group vehicles after 2000
Except the ones using Golf 7 (MQB) platform
Agenda

1. RKE from scratch
2. The VW systems
3. The Hitag2 system
The Hitag2 system

We verified our findings in practice by building a key emulator and then unlocking and locking the vehicles with newly generated rolling codes:

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Model</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alfa Romeo</td>
<td>Giulietta</td>
<td>2010</td>
</tr>
<tr>
<td>Chevrolet</td>
<td>Cruze Hatchback</td>
<td>2012</td>
</tr>
<tr>
<td>Citroen</td>
<td>Nemo</td>
<td>2009</td>
</tr>
<tr>
<td>Dacia</td>
<td>Logan II</td>
<td>2012</td>
</tr>
<tr>
<td>Fiat</td>
<td>Punto</td>
<td>2016</td>
</tr>
<tr>
<td>Ford</td>
<td>Ka</td>
<td>2009, 2016</td>
</tr>
<tr>
<td>Lancia</td>
<td>Delta</td>
<td>2009</td>
</tr>
<tr>
<td>Mitsubishi</td>
<td>Colt</td>
<td>2004</td>
</tr>
<tr>
<td>Nissan</td>
<td>Micra</td>
<td>2006</td>
</tr>
<tr>
<td>Opel</td>
<td>Vectra</td>
<td>2008</td>
</tr>
<tr>
<td>Opel</td>
<td>Combo</td>
<td>2016</td>
</tr>
<tr>
<td>Peugeot</td>
<td>207</td>
<td>2010</td>
</tr>
<tr>
<td>Peugeot</td>
<td>Boxer</td>
<td>2016</td>
</tr>
<tr>
<td>Renault</td>
<td>Clio</td>
<td>2011</td>
</tr>
<tr>
<td>Renault</td>
<td>Master</td>
<td>2011</td>
</tr>
</tbody>
</table>

The vehicles in the above list are our own and also from colleagues and friends who volunteered.
Analysis: black box reverse-engineering
Analysis: bitstreams

```
00 01 53 d0 11 6b 21 b1 d1 d2 3b e6 b7
00 01 53 d0 11 6b 21 b7 d1 df 62 16 15
00 01 53 d0 11 6b 21 bb f3 96 2d 8e a5
00 01 53 d0 11 6b 21 bd f2 9a 77 3a 40
00 01 53 d0 11 6b 21 c3 d5 c6 57 22 7d
00 01 53 d0 11 6b 21 c5 d5 ce 2b 22 0f
00 01 53 d0 11 6b 21 c9 f5 4b d5 ee 94
00 01 53 d0 11 6b 21 cf f5 c6 c5 0a eb
00 01 53 d0 11 6b 21 d3 75 ee 77 7e 99
00 01 53 d0 11 6b 21 d5 75 e7 15 92 18
00 01 53 d0 11 6b 21 d8 55 f3 fb 2a 77
00 01 53 d0 11 6b 21 de 55 f7 c1 de bb
00 01 53 d0 11 6b 21 e3 71 d8 6e 06 fa
00 01 53 d0 11 6b 21 e5 75 dd 3d ca 62
00 01 53 d0 11 6b 21 e8 55 5b 24 76 6c
00 01 53 d0 11 6b 21 ee 51 d4 76 fa 3f
00 01 53 d0 11 6b 21 f3 d1 d7 31 62 7e
00 01 53 d0 11 6b 21 f5 d1 d4 71 d6 8f
00 01 53 d0 11 6b 21 f9 f1 cf 86 9a 03
00 01 53 d0 11 6b 21 ff f0 ce ed 36 c2
```
Frequency analysis: changes

Computed over $> 2^{12}$ rolling codes from the same remote

→ First 51 bits don’t change on this remote
Frequency analysis: closer look

Exponential increase

→ Counter!
Blackbox analysis

\[< \text{start}||UID||btn||ctr||ks||chk >\]

Unknown keystream \(ks\)
Checksum \(chk\): XOR all other bytes

We need some external information on how \(ks\) is computed
A closer look at the remote

Philips Semiconductors

Product Profile

Security Transponder plus Remote Keyless Entry, **HITAG2PLUS**

**Features**
- Compatible with Security Transponder, PCF7936AS.
- Rolling Code Generator for keyless entry
- 14-pin SO package

**Transponder**
- 64/32 bit mutual authentication
- 32 bit unique device identification number
- Fast authentication, 39ms
- 40-bit Smart Key

**General Description**
The HITAG2PLUS is a high performance monolithic Security Transponder and Remote Keyless Entry Chip ideally suited for car immobiliser applications that incorporate keyless entry functions.

The HITAG2PLUS transponder circuitry is compatible with the Security Transponder PCF7936AS to support mixed systems using a HITAG2PLUS and a standard Security Transponder, PCF7936AS at the same time.

The Transponder circuitry meets the security and
Hitag2

Stream cipher used in transponders

\[ \text{Hitag2}(\text{key} = (48 \text{ bits}), \text{serial} = (32 \text{ bits}), \text{iv} = (32 \text{ bits})) \]

Reverse engineered, several attacks to recover the key from outputs

Re: frequency analysis of remote outputs
Hitag2 is used, so:

\[ ks = ???( \text{Hitag2(key = ???, serial = ???, iv = ???)} ) \]

\textit{UID} is 32 bits, just like the \textit{serial}…

\[ ks = ???( \text{Hitag2(key = ???, serial = } UID, \text{iv = ???)} ) \]

But what to use for the \textit{key}?
Security Transponder plus Remote Keyless Entry, HITAG2PLUS

**Features**
- Compatible with Security Transponder, PCF7936AS.
- Rolling Code Generator for keyless entry
- 14-pin SO package

**Transponder**
- 64/32 bit mutual authentication
- 32 bit unique device identification number
- Fast authentication, 39ms
- 40-bit Smart Key

**General Description**

The HITAG2PLUS is a high performance monolithic Security Transponder and Remote Keyless Entry Chip ideally suited for car immobiliser applications that incorporate keyless entry functions.

The HITAG2PLUS transponder circuitry is compatible with the Security Transponder PCF7936AS to support mixed systems using a HITAG2PLUS and a standard Security Transponder, PCF7936AS at the same time.

The Transponder circuitry meets the security and
### Transponder Memory

<table>
<thead>
<tr>
<th>Identifier</th>
<th>P0</th>
<th>53D0116B</th>
<th>R</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSW / ISK low</td>
<td>P1</td>
<td>4D494B52</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>NA / ISK high</td>
<td>P2</td>
<td>00004F4E</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>Configuration</td>
<td>P3</td>
<td>00AA4854</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>User page 0</td>
<td>P4</td>
<td>4D494B52</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>User page 1</td>
<td>P5</td>
<td>00004F4E</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>User page 2</td>
<td>P6</td>
<td>6515F2D5</td>
<td>R</td>
<td>W</td>
</tr>
<tr>
<td>User page 3</td>
<td>P7</td>
<td>00000000</td>
<td>R</td>
<td>W</td>
</tr>
</tbody>
</table>
Test case for Hitag2

Same value as in the test case of the Hitag2 code:

```
// "MIKRON" = O N M I K R
// Key = 4F 4E 4D 49 4B 52 - Secret 48-bit key
// Serial = 49 43 57 69 - Serial number
// Random = 65 6E 45 72 - Random IV
state = hitag2_init(rev64(0x524B494D4FUL), rev32(0x69574349),
                    rev32(0x72456E65));
```

→ That was a blank remote
Progress so far

Assuming we have everything right:

\[ ks = ???( \text{Hitag2}(\text{key} = 0x4f4e4d494b52, \text{serial} = UID, \text{iv} = ???) ) \]

Remember Hitag2 output is not random? → Generate random ivs
Progress so far

Confident that we are right:

\[ ks = \text{???}(\text{Hitag2(key = 0x4f4e4d494b52, serial = UID, iv = ???)}) \]

Even better: no output mangling:

\[ ks = \text{Hitag2(key = 0x4f4e4d494b52, serial = UID, iv = ???)} \]

\[ \rightarrow \text{All is left is the } iv \]
Finding the IV

There is some function \( \varphi \) so that \( iv = \varphi(key, UID, btn, ctr) \)

Assume that \( iv = \varphi(btn, ctr) \)

Assume linear: \( \varphi(btn, ctr) = \alpha.btn + \beta.ctr + \gamma \)

Set \( ctr = 0 \) (4 of our \( 2^{12} \) rolling codes since \( ctr \) is 10 bits long)

\[ \Rightarrow \varphi(btn, 0) = \alpha.btn + \gamma \]

Bruteforce \( \alpha \) and \( \gamma \)

We have hits for \( btn = 2, \alpha = 1, \gamma \in \{0x4000, 0x8000, 0xc000, 0x10000\} \)

\( \rightarrow \) Success \( \text{\o/} \)
Finding the IV

\[ \varphi(btn, ctr) = btn + \beta.ctr + \gamma, \gamma \in \{0x4000, 0x8000, 0xc000, 0x10000\} \]

Now use arbitrary \( ctr \), bruteforcing \( \beta \)
Gives \( \beta = 0x10 \)

Why \( \gamma \in \{0x4000, 0x8000, 0xc000, 0x10000\} \)\?
\( \rightarrow \) \( ctr \) is in fact more than 10 bits!

\[ \varphi(btn, ctr) = btn + 0x10.ctr \]

\[ \varphi(btn, ctr) = ctr || btn \]

\( \rightarrow \) We know everything!
Hitag2 scheme

\[
\begin{align*}
&\left\langle \text{start} || \text{UID} || \text{btn} || \text{ctr} \& \text{0x3ff} || \text{ks} || \text{chk} \right> \\
&\text{ks} = \text{Hitag2} (\text{key} = \text{key}_{\text{UID}}, \text{serial} = \text{UID}, \text{iv} = \text{ctr} || \text{btn})_{[0:31]} \\
&\text{chk} = \text{xor} (\text{other bytes})
\end{align*}
\]

In our case: \(\text{key}_{\text{UID}} = \text{0x4f4e4d494b52}\) (default Hitag2 key)

But this is the shared secret key of that remote

→ Contrary to VW schemes, good key diversification
Hitag2 scheme: attacks

To clone the remote, we need the $UID$, $ctr$ and the $key_{UID}$

Need 1 rolling code to get the $UID$ and $ctr$
Cannot recover $key_{UID}$ (48 bits) from less than 2 rolling codes (32 bits $ks$)

Attack in the paper:
- 4 to 8 rolling codes
- 10min on a laptop
How to get enough rolling codes

Wait enough time eavesdropping...

How to get rolling codes quicker?

\[
< \text{start} | \text{UID} | \text{btn} | \text{ctr} | \text{ks} | \text{chk} >
\]

Send noise on $\text{chk}$, the car ignore that rolling code
→ Owner presses button again
→ Or not

*Attacker wins in both cases!*
Conclusion

Agenda:
1. RKE from scratch
2. The VW systems
3. The Hitag2 system

Responsible disclosure:
- VW Group Dec 2015
- NXP Semiconductors Jan 2016

Poor crypto is bad, poor key management is worse
Weak ciphers still used in new vehicles
Improvement in ciphers over time (VW-x)
This may explain several mysterious theft cases without signs of forced entry

Q&A