Dismantling Megamos Crypto: Wirelessly Lockpicking a Vehicle Immobilizer

Roel Verdult
Radboud University
Nijmegen, NL

Flavio D. Garcia
University of
Birmingham, UK

Baris Ege
Radboud University
Nijmegen, NL
Why this special paper presentation?

• This paper was first accepted at Usenix Security’13
• VW sought an injunction from the High Court of London to prevent publication
• The High Court of London granted an interim injunction and therefore we had to withdraw the article
• We have now reached an amicable settlement without any admission of liability
• We will talk about the technical content of the paper but not about the details of the case
Vehicle Immobilizers

- Passive RFID Tags (125 KHz)
- Prevent hot-wiring
- Mandatory
  - Europe (EU Directive 95/56/EC)
  - Australia (AS/NZS 4601:1999)
  - Canada (CAN/ULC S338-98)
- Prevent insurance fraud
- Should not be confused with remote controls that unlock the car doors (433 MHz)
Vehicle Immobilizers
Three main immobilizer chips used (2012-13)

- TI’s DST (40-bit key)
  - Bono et al. “Security Analysis of a Cryptographically-Enabled RFID Device” [Usenix Security’05]

- NXP’s Hitag2 (48-bit key)
  - Analysed in our paper “Gone in 360 Seconds: Hijacking with Hitag2” [Usenix Security’12]

- EM’s Megamos Crypto (96-bit key)
  - This talk
### Megamos Crypto Usage (2013)

<table>
<thead>
<tr>
<th>Make</th>
<th>Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alfa Romeo</td>
<td>147, 156, GT</td>
</tr>
<tr>
<td>Buick</td>
<td>Regal</td>
</tr>
<tr>
<td>Cadillac</td>
<td>CTS-V, SRX</td>
</tr>
<tr>
<td>Chevrolet</td>
<td>Aveo, Kalos, Matiz, Nubira, Spark, Evanda, Tacuma</td>
</tr>
<tr>
<td>Citroen</td>
<td>Jumper (2008), Relay</td>
</tr>
<tr>
<td>Daewoo</td>
<td>Kalos, Lanos, Leganza, Matiz, Nubira, Tacuma</td>
</tr>
<tr>
<td>DAF</td>
<td>CF, LF, XF</td>
</tr>
<tr>
<td>Ferrari</td>
<td>California, 612 Schaglletti</td>
</tr>
<tr>
<td>Holden</td>
<td>Barina, Frontera</td>
</tr>
<tr>
<td>Isuzu</td>
<td>Rodeo</td>
</tr>
<tr>
<td>Iveco</td>
<td>Eurocargo, Daily</td>
</tr>
<tr>
<td>Kia</td>
<td>Carnival, Clarus, Pride, Shuma, Sportage</td>
</tr>
<tr>
<td>Lancia</td>
<td>Lybra, Musa, Thesis, Y</td>
</tr>
<tr>
<td>Maserati</td>
<td>Quattroporte</td>
</tr>
<tr>
<td>Opel</td>
<td>Frontera</td>
</tr>
<tr>
<td>Pontiac</td>
<td>G3</td>
</tr>
<tr>
<td>Porsche</td>
<td>911, 968, Boxster</td>
</tr>
<tr>
<td>Seat</td>
<td>Altea, Cordoba, Ibiza (2014), Leon, Toledo</td>
</tr>
<tr>
<td>Skoda</td>
<td>Fabia (2011), Felicia, Octavia, Roomster, Super, Yeti</td>
</tr>
<tr>
<td>Ssangyong</td>
<td>Kormando, Musso, Rexton</td>
</tr>
<tr>
<td>Tagaz</td>
<td>Road Partner</td>
</tr>
<tr>
<td>Volvo</td>
<td>C30, S40 (2005), S60, S80, V50 (2005), V70, XC70, XC90, XC94</td>
</tr>
</tbody>
</table>
Hardware Setup

- Proxmark III
  - 125 kHz, 13.56 MHz
  - ADC, uC and FPGA
  - Open design/source
## Tag Memory layout
(from datasheet)

<table>
<thead>
<tr>
<th>Block</th>
<th>Content</th>
<th>Denoted by</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>user memory</td>
<td>$um_0$ $um_{15}$</td>
</tr>
<tr>
<td>1</td>
<td>user memory, lock bits</td>
<td>$um_{16}$ $um_{29l_0l_1}$</td>
</tr>
<tr>
<td>2</td>
<td>device identification</td>
<td>$id_0$ $id_{15}$</td>
</tr>
<tr>
<td>3</td>
<td>device identification</td>
<td>$id_{16}$ $id_{31}$</td>
</tr>
<tr>
<td>4</td>
<td>crypto key</td>
<td>$k_0$ $k_{15}$</td>
</tr>
<tr>
<td>5</td>
<td>crypto key</td>
<td>$k_{16}$ $k_{31}$</td>
</tr>
<tr>
<td>6</td>
<td>crypto key</td>
<td>$k_{32}$ $k_{47}$</td>
</tr>
<tr>
<td>7</td>
<td>crypto key</td>
<td>$k_{48}$ $k_{63}$</td>
</tr>
<tr>
<td>8</td>
<td>crypto key</td>
<td>$k_{64}$ $k_{79}$</td>
</tr>
<tr>
<td>9</td>
<td>crypto key</td>
<td>$k_{80}$ $k_{95}$</td>
</tr>
<tr>
<td>10</td>
<td>pin code</td>
<td>$pin_0$ $pin_{15}$</td>
</tr>
<tr>
<td>11</td>
<td>pin code</td>
<td>$pin_{16}$ $pin_{31}$</td>
</tr>
<tr>
<td>12</td>
<td>user memory</td>
<td>$um_{30}$ $um_{45}$</td>
</tr>
<tr>
<td>13</td>
<td>user memory</td>
<td>$um_{46}$ $um_{61}$</td>
</tr>
<tr>
<td>14</td>
<td>user memory</td>
<td>$um_{62}$ $um_{77}$</td>
</tr>
<tr>
<td>15</td>
<td>user memory</td>
<td>$um_{78}$ $um_{93}$</td>
</tr>
</tbody>
</table>
**Megamos Authentication Protocol**

\[
\begin{align*}
\text{id} & \quad \xrightarrow{\text{id}} \quad \text{id} \\
\text{n}_C, \text{a}_C & \quad \xrightarrow{\text{n}_C, \text{a}_C} \quad \text{a}_T \\
\end{align*}
\]

- \text{id} = 32-bit Tag identifier
- \text{n}_C = 56-bit Car nonce
- \text{a}_C = 28-bit Car authenticator (keystream)
- \text{a}_T = 20-bit Tag authenticator (keystream)
Reverse engineering Megamos Crypto

• We discovered that the Tango Programmer (car diagnostic tool) uses the Megamos Crypto algorithm since 2009 (for testing purposes only)
• We reverse-engineered the algorithm from the freely available Tango software package bypassing its obfuscation.
... but you can also read it directly from the car’s ECU

NEC uPD78P083 has simply no protection
Cryptanalysis - Pre-requisites

- Requires access to the car and the car key
- Adversary needs to turn the ignition on twice and eavesdrop two traces

<table>
<thead>
<tr>
<th>Origin</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car Transponder</td>
<td>AA AA AA AA AA AA AA</td>
</tr>
<tr>
<td>Car Transponder</td>
<td>60 9D 6</td>
</tr>
<tr>
<td>Car Transponder</td>
<td>59 85 95 13</td>
</tr>
<tr>
<td>Car Transponder</td>
<td>3F 3F 1F 1F B6 CC 51 3F</td>
</tr>
<tr>
<td>Car Transponder</td>
<td>30 00 95 13</td>
</tr>
<tr>
<td>Car Transponder</td>
<td>5 08 4D EC</td>
</tr>
<tr>
<td>Car Transponder</td>
<td>3</td>
</tr>
</tbody>
</table>
Complexity analysis of the cipher
The Megamos Crypto Cipher

Secret key size = 96 bits
Internal state size = 23 + 13 + 3x7 = 57 bits
Megamos Crypto Initialization and workings

\[ k_{95} \ldots k_{40} + \text{nonce } n_c \]

\[
\begin{align*}
&k_{39} \ldots k_0 \\
&0 \ldots 0 \\
&s_0 \ldots s_1 \\
&s_7 \\
&s_{35} \ldots s_{40} \\
&s_{55}
\end{align*}
\]

\[ n_C = 56\text{-bit Car nonce} \]
\[ a_C = 28\text{-bit Car authenticator (keystream)} \]
\[ a_T = 20\text{-bit Tag authenticator (keystream)} \]
Cryptanalysis of Megamos Crypto

- Take the first authentication trace
- Trying all $2^{56}$ states $s_{40}$, running 15 steps discarding on the output leaves $2^{41}$ candidate states
- After running the cipher backwards to $s_7$ we still have $2^{41}$ candidates
- Running backwards to $s_0$ guessing 7 bit leaves $2^{48}$ candidate keys.
- Check against a second authentication trace singles out the key.
Cryptanalysis of Megamos Crypto

• Total attack complexity reduced from $2^{96}$ to less than $2^{56}$ encryptions
• Takes less than two days on a Copacobana‘05
• This complexity can be further reduced by precomputation:
  – E.g., using a 12 Terabyte table reduces the complexity to $2^{49}$ table lookups
  – This has some practical limitations
Partial Key-update Attack

Observations:

During our research, the majority of deployed tags we found were:

- Unlocked $l_0 = 0$ (writable)
- Could be unlocked with a default PIN code

- The 96-bit secret key is written to the tag in blocks of 16 bits instead of being an atomic operation.

<table>
<thead>
<tr>
<th>Block</th>
<th>Content</th>
<th>Denoted by</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>user memory</td>
<td>$um_0$-$um_{15}$</td>
</tr>
<tr>
<td>1</td>
<td>user memory, lock bits</td>
<td>$um_{16}$-$um_{29} l_1$</td>
</tr>
<tr>
<td>2</td>
<td>device identification</td>
<td>$id_0$-$id_{15}$</td>
</tr>
<tr>
<td>3</td>
<td>device identification</td>
<td>$id_{16}$-$id_{31}$</td>
</tr>
<tr>
<td>4</td>
<td>crypto key</td>
<td>$k_0$-$k_{15}$</td>
</tr>
<tr>
<td>5</td>
<td>crypto key</td>
<td>$k_{16}$-$k_{31}$</td>
</tr>
<tr>
<td>6</td>
<td>crypto key</td>
<td>$k_{32}$-$k_{47}$</td>
</tr>
<tr>
<td>7</td>
<td>crypto key</td>
<td>$k_{48}$-$k_{63}$</td>
</tr>
<tr>
<td>8</td>
<td>crypto key</td>
<td>$k_{64}$-$k_{79}$</td>
</tr>
<tr>
<td>9</td>
<td>crypto key</td>
<td>$k_{80}$-$k_{95}$</td>
</tr>
<tr>
<td>10</td>
<td>pin code</td>
<td>$pin_0$-$pin_{15}$</td>
</tr>
<tr>
<td>11</td>
<td>pin code</td>
<td>$pin_{16}$-$pin_{31}$</td>
</tr>
<tr>
<td>12</td>
<td>user memory</td>
<td>$um_{30}$-$um_{45}$</td>
</tr>
<tr>
<td>13</td>
<td>user memory</td>
<td>$um_{46}$-$um_{61}$</td>
</tr>
<tr>
<td>14</td>
<td>user memory</td>
<td>$um_{62}$-$um_{77}$</td>
</tr>
<tr>
<td>15</td>
<td>user memory</td>
<td>$um_{78}$-$um_{93}$</td>
</tr>
</tbody>
</table>
Partial Key-update Attack (simple)

- Get one authentication attempt from the car
- Guess 16 bits, write on one block then authenticate to the tag.
- If it succeeds you learn 16 key bits.
- This requires $6 \times 2^{16}$ writes and authenticate.
- Takes 25’ per block $\approx 2.5$ hours in total, using a Proxmark
Partial Key-update Attack (optimized)

- Same principle but only write zeros once in the first block
- Then increment the nonce and authenticate until the tag accepts
  - Remember key is added to the nonce during initialisation
- Repeat for another two blocks then combine with the cryptanalytic attack searching for the remaining bits
- This attack requires 6 writes and $3 \times 2^{16}$ authentications with the tag and negligible computational complexity
- The whole attack takes <30 minutes using a Proxmark III
Immobilizer Demo
Weak key attack

Some interesting keys we found

• If the key starts with 32 zero bits then you can use a time-memory trade-off as in [Oechslin’03]
• Build (once) a 1.5 Terabyte rainbow table (less than one week to build)
• Computational complexity of $2^{37}$ encryptions
• Few minutes computation on a laptop
Weak key attack

Some even more interesting keys we found

<table>
<thead>
<tr>
<th>Car</th>
<th>Secret key</th>
</tr>
</thead>
<tbody>
<tr>
<td>A 1</td>
<td>000000000d8 b3967c5a3c3b29</td>
</tr>
<tr>
<td>A 2</td>
<td>000000000d9 b79d7a5b3c3b28</td>
</tr>
<tr>
<td>B 1</td>
<td>00000000000 00010405050905</td>
</tr>
</tbody>
</table>

- These keys appear to have at most 32 bits of entropy
- An exhaustive search on such key takes only seconds
Mitigation and Alternatives

• Car owners can set lock-bit $l_0$ to one, set a random PIN. This prevents our partial key update attack.
• Set full entropy keys (locksmiths, dealers)
• Vehicle immobilizer tags based on the Advanced Encryption Standard (AES)
  • HITAG Pro, NXP Semiconductors (2007)
  • ATA5580, Atmel Corporation (2010)
  • TRPWS21/TRPBS27, Texas Instruments (2010)
Atmel Open Immobiliser Protocol Stack

• Atmel Corporation states in the datasheets:
  “Rather than developing its own proprietary cryptographic functions, Atmel selected and implemented the 128-bit AES-128 global benchmark standard as its data encryption and decryption source. This open source standard is freely available to the public for use and scrutiny. Because of this it continues to be favored by industry experts over private and proprietary crypto algorithms.”

• Key Features
  – No security by obscurity
  – Use of 128-bits AES
  – Car & key send challenge
  – Open protocol design
  – Open source examples
  – Allows public evaluation
Responsible disclosure

- We carefully followed the official guidelines from the Dutch Government [1]
- We notified the chip manufacturer in November 2012, nine months ahead of scheduled publication at Usenix’13.
- We invested many days to inform them properly
  - conference call
  - several letters and emails
  - personal meeting
- We understand that measures have been taken to prevent our weak-key and partial key-update attacks in newer vehicles

Thanks for staying around!

Acknowledgements

We would like to thank the following colleagues and friends for their firm support (in alphabetical order)

Ross Anderson          Sam King
Robert Carolina       Bas Kortmann
Tom Chothia           Kenny Paterson
Riccardo Focardi      Carolyn Pike
Dorine Gebbink        Jon Rowe
Casey Henderson       Mark Ryan
Bart Jacobs           Graham Steel