Dismantling Megamos Crypto: Wirelessly Lockpicking a Vehicle Immobilizer

Roel Verdult
Radboud University Nijmegen, The Netherlands
rverdult@cs.ru.nl

Baris Ege
Radboud University Nijmegen, The Netherlands

Flavio D. Garcia
School of Computer Science, University of Birmingham, UK.

Institute for Computing and Information Sciences
Radboud University Nijmegen, The Netherlands
rverdult@cs.ru.nl
www.cs.ru.nl/~rverdult
Disclaimer

• Due to a recent injunction by the High Court of London this talk cannot cover the technical core of the accepted paper

We are responsible for the content of this presentation and any opinions expressed during the presentation are ours and do not necessarily represent the views of the University of Birmingham or Dr Garcia
Interaction

• We stick to these slides and will not answer questions (sorry about this; we hope you understand)

• However, we are happy to see discussion about these issues in the community

• Since legal procedures are still ongoing, it is not appropriate to go into details
Contents

• Introduction
  – Vehicle Immobilizers

• Related Work
  – Related work on vehicle immobilizer systems
  – Weaknesses in Hitag2 cryptography
  – Responsible disclosure
  – Reverse Engineering of security mechanisms

• Mitigation and alternatives
  – Proposed in the academic literature
  – Introduced by the industry

• Conclusion
Vehicle Immobilizers

- Electronic anti-theft device
- Introduced in the '90s
- Prevents hot-wiring (like in the movies)
- Mandatory in many countries eg.
  - Europe (EU Directive 95/56/EC)
  - Australia (AS/NZS 4601:1999)
  - Canada (CAN/ULC S338-98)
- Passive RFID transponder (125 KHz)
- **Not** to be confused with remote controls that unlock the car doors (433/868 MHz)
**Vehicle Immobilizer Example**

Hello, tell me your number

Key number (identifier)

Random number + Proof_{car}

Load random number, only respond to valid Proof_{car}

Proceed *only* when identifier is *known*

Start the engine *only* when Proof_{key} is *valid*
Related Work (DST)

- Digital Signature Transponder (DST)
  - Introduced in 1995
  - Key length of 40-bits
  - Complexity $2^{40}$ encryptions (exhaustive search)

- Publications

- Replaced by DST80 or TRPWS21 (AES 128-Bit)
Related Work (KeeLoq)

- **KeeLoq**
  - Introduced in 1996
  - Key length of 64-bits
  - Complexity $2^{44}$ encryptions (slide / meet-in-the-middle)

- **Publications**

- **No longer deployed in the automotive industry**
Related Work (Hitag2)

- **Hitag2**
  - Introduced in 1996
  - Key length of 48-bits
  - Complexity $2^{37}$ encryptions (cryptanalytic)

- **Publications**

- **Replaced by Hitag3 or Hitag-Pro (AES 128-Bit)**
Identified Hitag2 Weaknesses

• Weak cryptographic algorithm
  – Cipher design is obsolete (from the early ’90s)
  – Weak and invertible cipher initialization
  – Security is significantly lower than the key size

• Implementation mistakes
  – There is no random number (freshness) introduced by the transponder during authentication
  – Secret key update is not one atomic operation

• Improper usage by car manufacturers
  – Many transponders are configured with the default (or easy to guess) passwords
  – Many cars are configured with weak secret keys which drastically speeds-up key recovery
Practicality of Hitag2 Attacks

1. Communicate with the genuine car-key
   - With maximum wireless distance of two inches

2. Bypass other security measures of the car
   - Force the door locks of the car
   - Disable the alarm (separate protection)
   - Force the ignition lock (hot-wire the car)

3. Eavesdrop immobilizer messages from the car

4. Communicate again with the car-key

5. Perform a complex mathematical computation to recover the secret cryptographic key

6. Emulate the car key and start the car
This is not really a very practical way to steal a car!
Megamos Crypto

- Introduced in 1995/1996
- Key length of 96-bits

- Security issues are comparable to the weaknesses we found in our Hitag2 study

- Practicality of attacks are similar to Hitag2

- Still deployed in cars
Responsible disclosure in general

- Notify "problem owner" early on
- Help to understand/solve issues
- Publish after a delay
  - CERT - Carnegie Mellon 45 days
    [http://www.cert.org/kb/vul_disclosure.html](http://www.cert.org/kb/vul_disclosure.html)
  - Google recommends 60 days
    [http://googleonlinesecurity.blogspot.nl/2013/05/disclosure-timeline-for-vulnerabilities.html](http://googleonlinesecurity.blogspot.nl/2013/05/disclosure-timeline-for-vulnerabilities.html)
  - Guidelines from the Dutch government
  - 60 days for software
  - 6 months for hardware
Nijmegen’s History

• Responsible disclosure
• NXP MIFARE Classic chip
  – contested in court by NXP, but they lost
  – published after 6 months
• Nokia NFC cell-phones after 6 months
• Atmel's crypto memory after 6 months
• NXP Hitag2 after 6 months
• In these cases, the notified parties contacted us within 48 hours
Notification Chains

• Previous notified parties explicitly demanded that they should communicate with their customers
• We should not talk to them directly
  – Such notification is their responsibility
  – They do not share customer information
  – They know which people to contact
• This is what we have done since then
Megamos Notification

- Notified manufacturer **twice** in 2012
  - Through email and registered mail

Further collaboration
- Conference call
- Letters and emails
- Personal meeting
- Constructive discussions

No contacts at this level
Reverse Engineering Methods

• Observing in/output (blackbox)
  – Non-invasive, can be performed remotely
• Decompile firmware (software)
  – Automatic tools are available to assist
• Chip slicing
  – Peeling away layers of a chip
  – Use microscope to shoot photos
• New approach: Side Channel Analysis for Reverse Engineering (SCARE)
Reverse Engineering – Observing in/output

• Publications that used blackbox reverse-engineering

<table>
<thead>
<tr>
<th>LFSR \ XX</th>
<th>55</th>
<th>54</th>
<th>51</th>
<th>50</th>
<th>45</th>
<th>44</th>
<th>41</th>
<th>40</th>
<th>15</th>
<th>14</th>
<th>11</th>
<th>10</th>
<th>05</th>
<th>04</th>
<th>01</th>
<th>00</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xb05d53bfdbXX</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0xfb57bbc7fXX</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0xe2fd86e299XX</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Reverse Engineering – Firmware (software)

• Publications about reversing security algorithms

• Miller & Valasek at DEFCON 2013 (funded by DARPA)
  − http://illmatics.com/car_hacking.pdf
Reverse Engineering – Chip slicing

• Publications about chip slicing


• Pictures are published at [http://www.siliconzoo.org](http://www.siliconzoo.org)
Access to Megamos Crypto

• We used Tango Programmer, whose software is publicly available
  – Contains Megamos Crypto algorithm obfuscated
  – Called “murky” by the judge
  – Available since 2009 and still being distributed

• There seem to be many other sources available which contain the algorithm and are
  – easy accessible
  – not protected
  – not encrypted
  – not obfuscated
Megamos Crypto partly published

• Challenges in Security Engineering event
  – CSE 2012 3-7 Sept. 2012 Bochum, Germany
  – ECRYPT II Summer School
  – European Network of Excellence for Cryptology II
    • Funded by European Commission's Seventh Framework Programme (FP7)

• Jan Krissler (Starbug) presented at CSE 2012
  – Reverse engineering Megamos Crypto via chip slicing
  – Hi-resolution pictures of the hardware logic gates
  – Schematic of a core component of Megamos Crypto
  – Slides are publicly available for download
Mitigation and Alternatives

• Proposed in the academic literature
  – Authentication protocols
  – Key derivation schemes

• Products introduced by the industry
  – Proprietary chips (Hitag3 and DST80)
  – Vehicle immobilizer transponders based on the Advanced Encryption Standard (AES)
    • HITAG Pro, NXP Semiconductors (2007)
    • ATA5580, Atmel Corporation (2010)
    • TRPWS21/TRPBS27, Texas Instruments (2010)
Proposed in the academic literature


Atmel Open Immobilizer Protocol Stack

• Atmel Corporation states in their 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
Conclusion

• Automotive industry has focus on safety, but not enough on security

• They still use proprietary out-dated cryptographic algorithms

• Maybe they should consider a new update model (like “Patch Tuesday”)
Historical claim

Final paper SHA-512 hash:

9d05ba88740499eeceea3d8609174b444
43683da139f78b783666954ccc605da8
4601888134bf0c23ba46fb4a88c056bf
bbb629e1ddffcf60fa91880b4d5b4aca