CANAuth
Backward Compatible Authentication for CAN

Anthony Van Herreweghe
Dave Singelee, Ingrid Verbauwhede

21 October 2011
Outline

1. Why
2. How
3. CANAuth
4. Security

Anthony Van Herrewege et al.
CANAuth - Backward Compatible Authentication for CAN
<table>
<thead>
<tr>
<th>#</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Why</td>
</tr>
<tr>
<td>2</td>
<td>How</td>
</tr>
<tr>
<td>3</td>
<td>CANAuth</td>
</tr>
<tr>
<td>4</td>
<td>Security</td>
</tr>
</tbody>
</table>
Attacks

- Brakes
Attacks

- Brakes
- Automatic transmission
Attacks

- Brakes
- Automatic transmission
- Airconditioning
Attacks

- Brakes
- Automatic transmission
- Airconditioning
- ...
Prevention

- Encryption
Prevention

- Encryption: Automatic radio volume, OBD interface, ... $\rightarrow$ broken
Prevention

- Encryption:
  
  Automatic radio volume, OBD interface, . . .
  \[\rightarrow\] broken

- Authentication
Prevention

- Encryption:
  Automatic radio volume, OBD interface, ...

- Authentication:
  Existing features can still work, but ...
Restrictions

- Hard real-time → all data in message
Restrictions

- Hard real-time $\rightarrow$ all data in message
  
  no $(\mu)$TESLA
Restrictions

- Hard real-time → all data in message
  no $(\mu)$TESLA
- Limited message size → very lightweight
Restrictions

- Hard real-time → all data in message
  no \((\mu)\)TESLA

- Limited message size → very lightweight
  no solutions by Gennaro & Rohatgi
Restrictions

- Hard real-time → all data in message
  no \((\mu)\)TESLA
- Limited message size → very lightweight
  no solutions by Gennaro & Rohatgi
- Broadcast messages → no pairwise keys
Restrictions

- Hard real-time → all data in message
  - no $(\mu)$TESLA
- Limited message size → very lightweight
  - no solutions by Gennaro & Rohatgi
- Broadcast messages → no pairwise keys
- Limited IDs → no new message “types”
Restrictions

- Hard real-time → all data in message
  no \((\mu)\)TESLA

- Limited message size → very lightweight
  no solutions by Gennaro & Rohatgi

- Broadcast messages → no pairwise keys

- Limited IDs → no new message “types”

- Backward compatible → need out-of-band channel
Up to 15 extra bits data per CAN bit @ 1 MHz
- Up to 15 extra bits data per CAN bit @ 1 MHz
- Backward compatible out-of-band transmission
CAN+

- Up to 15 extra bits data per CAN bit @ 1 MHz
- Backward compatible out-of-band transmission
# Outline

1. Why
2. How
3. CANAuth
4. Security

---

Anthony Van Herrewege et al.
CANAuth - Backward Compatible Authentication for CAN
Frame Format

- Shortest message $= 1$ byte
Frame Format

- Shortest message = 1 byte
  \[ 8 \cdot 15 = 120 \text{ CAN+ bits} \]
Shortest message = 1 byte

⇒ 8 \cdot 15 = 120 \text{ CAN+ bits}
Key Establishment

- Group of messages $G_i \rightarrow$ pre-shared $K_{p_i}$
Key Establishment

- Group of messages $G_i \rightarrow$ pre-shared $\mathcal{K}_p_i$
- Transmitting node sends:

<table>
<thead>
<tr>
<th>Status bits</th>
<th>Counter value</th>
<th>Random Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>[2 + 6 bits]</td>
<td>[24 bits]</td>
<td>[88 bits]</td>
</tr>
</tbody>
</table>
Key Establishment

- Group of messages $G_i \rightarrow$ pre-shared $K_{P_i}$
- Transmitting node sends:

<table>
<thead>
<tr>
<th>10</th>
<th>0</th>
<th>$\text{ctrA}_i$</th>
<th>$r_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status bits</td>
<td>Counter value</td>
<td>Random Number</td>
<td></td>
</tr>
<tr>
<td>[2 + 6 bits]</td>
<td>[24 bits]</td>
<td>[88 bits]</td>
<td></td>
</tr>
</tbody>
</table>

- Only accepted if $\text{ctrA}_{i,\text{sent}} > \text{ctrA}_{i,\text{stored}}$
Key Establishment

- Group of messages $G_i \rightarrow$ pre-shared $K_{pi}$
- Transmitting node sends:

<table>
<thead>
<tr>
<th>10</th>
<th>0</th>
<th>$\text{ctrA}_i$</th>
<th>$r_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status bits</td>
<td>Counter value</td>
<td>Random Number</td>
<td></td>
</tr>
<tr>
<td>[2 + 6 bits]</td>
<td>[24 bits]</td>
<td>[88 bits]</td>
<td></td>
</tr>
</tbody>
</table>

- Only accepted if $\text{ctrA}_{i,\text{sent}} > \text{ctrA}_{i,\text{stored}}$
- Session key:

$$K_{si} = \text{HMAC}(K_{pi}, \text{ctrA}_i \parallel r_i) \mod 2^{128}$$
Check for errors and/or tampering:

<table>
<thead>
<tr>
<th>Status bits [2 + 6 bits]</th>
<th>Signature [112 bits]</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>0</td>
</tr>
</tbody>
</table>

Anthony Van Herrewege et al.
CANAuth - Backward Compatible Authentication for CAN
### Key Establishment - Continued

- **Check for errors and/or tampering:**

  \[
  \text{sig}_{A_i} = \text{HMAC}(K_{S_i}, \text{ctr}_{A_i} \mathrel{\|} r_i) \mod 2^{112}
  \]
Message Authentication

- **Attach signature:**

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>(\text{ctrM}_i)</th>
<th>(\text{sigM}_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status bits</td>
<td>Counter value</td>
<td>Signature</td>
<td></td>
</tr>
<tr>
<td>[1 + 7 bits]</td>
<td>[32 bits]</td>
<td>[80 bits]</td>
<td></td>
</tr>
</tbody>
</table>

\[\text{sigM}_i = \text{HMAC}(K_s, \text{ctrM}_i \parallel \text{msg}_i) \mod 2^{80}\]

Only accepted if \(\text{ctrM}_i\), sent, > \(\text{ctrM}_i\), stored
Message Authentication

- Attach signature:

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>ctrM&lt;sub&gt;i&lt;/sub&gt;</th>
<th>sigM&lt;sub&gt;i&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status bits</td>
<td>Counter value</td>
<td>Signature</td>
<td></td>
</tr>
<tr>
<td>[1 + 7 bits]</td>
<td>[32 bits]</td>
<td>[80 bits]</td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{sigM}_i = \text{HMAC}(K_{S_i}, \text{ctrM}_i \parallel \text{msg}_i) \mod 2^{80} \]
Message Authentication

- Attach signature:

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>ctrM&lt;sub&gt;i&lt;/sub&gt;</th>
<th>sigM&lt;sub&gt;i&lt;/sub&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status bits</td>
<td>Counter value</td>
<td>Signature</td>
<td></td>
</tr>
<tr>
<td>[1 + 7 bits]</td>
<td>[32 bits]</td>
<td>[80 bits]</td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{sigM}_i = \text{HMAC}(K_{s_i}, \text{ctrM}_i \parallel \text{msg}_i) \mod 2^{80} \]

- Only accepted if \( \text{ctrM}_{i,\text{sent}} > \text{ctrM}_{i,\text{stored}} \)
Outline

1. Why
2. How
3. CANAuth
4. Security

Anthony Van Herrewege et al.
CANAuth - Backward Compatible Authentication for CAN
Brute-force Attacks

- Signature is only 80 bits!
Brute-force Attacks

- Signature is only 80 bits!
- $2^{64}$ CAN messages of 1 byte at 1 MHz takes $3.4 \times 10^7$ year
Brute-force Attacks

- Signature is only 80 bits!
- $2^{64}$ CAN messages of 1 byte at 1 MHz takes $3.4 \times 10^7$ year
- In reality: a lot longer $\rightarrow$ should be safe
Replay Attacks

- Adversary $\mathcal{A}$ can replay messages for certain $K_{s_i}$ if:
Replay Attacks

- Adversary $A$ can replay messages for certain $K_{Si}$ if:

\[
\begin{align*}
\text{ctrA}_{i,sent} &> \text{ctrA}_{i,stored} \\
\text{ctrM}_{i,sent} &> \text{ctrM}_{i,stored}
\end{align*}
\]
Adversary $A$ can replay messages for certain $K_{s_i}$ if:

\[
\begin{cases}
\text{ctr}_{A_i,\text{sent}} > \text{ctr}_{A_i,\text{stored}} \\
\text{ctr}_{M_i,\text{sent}} > \text{ctr}_{M_i,\text{stored}}
\end{cases}
\]

However ...
Tamper/Replay Resistance

- $A$ replays messages or finds tuple $\{\text{ctrA}_i, r_i, K_{s_i}\}$
Tamper/Replay Resistance

- \( A \) replays messages or finds tuple \( \{ \text{ctr}A_i, r_i, Ks_i \} \)
- Receiving nodes increase \( \text{ctr}A_{i, stored} \)
Tamper/Replay Resistance

- $A$ replays messages or finds tuple $\{\text{ctr}A_i, r_i, K_{s_i}\}$
- Receiving nodes increase $\text{ctr}A_i,\text{stored}$
  \[ \Rightarrow \text{ctr}A_i,\text{receiver} \neq \text{ctr}A_i,\text{sender} \]
Tamper/Replay Resistance

- $A$ replays messages or finds tuple $\{\text{ctr}A_i, r_i, Ks_i\}$
- Receiving nodes increase $\text{ctr}A_{i,\text{stored}}$
  - $\Rightarrow \text{ctr}A_{i,\text{receiver}} \neq \text{ctr}A_{i,\text{sender}}$
  - $\Rightarrow$ machine broken