From: <[cbauer@ycp.edu](mailto:cbauer@ycp.edu)>  
Date: Tue, Mar 22, 2016 at 7:40 AM  
Subject: Cryptologia - Decision on Manuscript ID UCRY-2016-0011  
To: [jack.schmandt@gmail.com](mailto:jack.schmandt@gmail.com)  
  
  
22-Mar-2016  
  
Dear Mr Schmandt:  
  
I regret to inform you that our reviewers have now considered your paper but unfortunately feel it unsuitable for publication in Cryptologia.   For your information I attach the reviewer comments at the bottom of this email.  I hope you will find them to be constructive and helpful.  You are of course now free to submit the paper elsewhere should you choose to do so.  
  
Thank you for considering Cryptologia. I hope the outcome of this specific submission will not discourage you from the submission of future manuscripts.  
  
Sincerely,  
Craig Bauer  
Editor in Chief, Cryptologia  
[cbauer@ycp.edu](mailto:cbauer@ycp.edu)  
  
  
Reviewer(s)' Comments to Author:  
  
Reviewer: 1  
  
Comments to the Author  
Improving the security of automobiles is an important topic and the  
lack of authentication of messages sent over the CAN bus is a real  
weakness. As a result, any defense that meaningfully makes an  
attacker's job more difficult would be an improvement. Your proposed  
solution does not seem to be such a defense.  
  
The paper suffers from a number of weaknesses: the authors seem to  
misunderstand the CAN protocol, CAN transceiver hardware, and the  
nature of ECUs and the messages they send. In addition, Mini-MAC would  
be but one component of a complete story. Important steps such as key  
management/distribution and ECU replacement are not considered. I'll  
address these in turn in more detail.  
  
CAN protocol: The CAN protocol allows a variable-length message with a  
payload of 0 to 8 bytes. One of the major claims in the paper is that  
Mini-MAC does not take any additional bits (e.g., Table 1). But this  
is false. It appears to add 0 to 32 bits (possibly 64 bits; it's not  
completely clear if you ever use more than 4-byte authentication tags)  
to each message. This would definitely have the effect of delaying  
messages as other ECUs would have to wait for longer messages,  
contrary to the claim in the paper.  
  
CAN transceivers: In Section 9.2, you suggest repurposing the CRC  
field in the CAN frame to add additional bytes to the MAC. You say  
that this could be implemented by modifying low-level code in the CAN  
stack. This doesn't work. The CRC is added by the transceiver hardware  
and is not under the control of software unless CAN is implemented via  
bit-banging--which it isn't in real ECUs. Even if you were to create  
new CAN hardware that allowed replacing the CRC field, you couldn't  
use it with standard CAN hardware (as is federally mandated in the US)  
because a standard receiver would detect a CRC error and send an error  
frame (which would cause every other receiver to send an error frame).  
  
ECUs: ECUs support significant functionality beyond that you would  
normally see. For example, Koscher et al. report that their cars' ECUs  
respond to DeviceControl packets that can perform dangerous  
operations. These packets (in addition to others described) are long,  
often 8 bytes. Mini-MAC couldn't protect those. Beyond that, there are  
often higher-level protocols that are built on top of CAN frames that  
support longer messages. For example, GMLAN supports longer messages  
by sending multiple CAN frames. Adding a MAC to the higher-level  
messages may make more sense than to the CAN frames.  
  
You say that ECUs should generate their own keys. There's no way that  
is going to work. Checkoway et al. report that their cars' telematics  
ECU--one of the most computationally powerful ECUs in the car--has its  
random number generator for authentication seeded with the same value  
each time due to a lack of entropy. Less-capable ECUs are going to be  
in an even worse position to generate keys. This is a common problem  
with embedded systems.  
  
You're also assuming that every ECU has writable, nonvolatile storage  
in which it can store counter values or previous messages. This seems  
unrealistic.  
  
One of your claims is that this does not add much of a computational  
burden to ECUs, but I don't think that's correct. Generally, CAN  
hardware can be programmed to cause an interrupt when a frame with an  
ID that matches a filter is received. (Or an interrupt occurs on every  
frame and nonmatching IDs cause the hardware to go back to sleep  
immediately.) With Mini-MAC, every device on the network needs to wake  
up, processes the message, and store it for future use, even if it  
doesn't care about the message at all.  
  
Different ECUs take different amounts of time to boot up but some send  
messages immediately. (E.g., the VIN seems to be sent in a few CAN  
frames when one of the ECUs turns out.) Any ECU which hasn't booted  
and initialized its CAN hardware will miss a message, leading to  
desynchronization.  
  
Complete solution: A complete solution designed to authenticate  
messages using a MAC would necessarily entail a description of how  
keys would be generated/distributed. As noted above, ECUs cannot  
generate their own keying material. What happens with a  
mechanic/dealer replaces an ECU? How do shop tools communicate with  
ECUs unless they have the MAC keys? How do they reprogram them? What  
prevents a compromised ECU from reprogramming other ECUs following the  
same mechanism?