

ADvanced Encryption STandard (ADESTA) for Diagnostics over CAN

Jihas Khan
Tata Elxsi, Ltd.

ABSTRACT

Security access feature based on seed-key mechanism is widely used in automotive electronics, mainly for flashing ECU software, writing or reading specific parameter values and running diagnostic routines. There exist a number of techniques to decode the algorithm for key generation from a specific seed. Such techniques can put vehicle network at great risks due to an intruder flashing unauthorized version of ECU software, or changing internal parameters of ECU, or changing a VIN number. A lot more similar malicious attacks can be done by getting control over the ECUs. Attackers can exploit this vulnerability to alter the performance from the stock and affect the safety of the passengers. A novel and fool proof algorithm to protect the vehicle and ECU from such malicious attacks is explained in this paper. An advanced encryption technique is developed and tested in ECU to replace the current seed-key mechanisms for ECU security guarantying a secure operation of the vehicle. Algorithm is implemented in the presentation layer of the OSI model which would ensure that the encryption details are not available to users and the ECU can be electronically accessed only by intended individuals.

CITATION: Khan, J., "ADvanced Encryption STandard (ADESTA) for Diagnostics over CAN," *SAE Int. J. Passeng. Cars – Electron. Electr. Syst.* 8(2):2015, doi:10.4271/2015-01-0206.

INTRODUCTION

Diagnostic system requirements are established in ISO 14229 for any serial data link. This International Standard specifies requirements of diagnostic services, which would allow a client (Example: A diagnostic tool possessed by service engineer) to control the diagnostic functions in an ECU or server (Example: Body Control Module) connected on a serial data link like CAN. This section of the paper provides an overview of the security access feature of ISO 14229 along with the OEM specific algorithms, its necessity and method of implementation.

Certain features of ECU like access to specific data and/or diagnostic services are provided with restricted access for security, emissions and safety reasons. Some cases where such a restriction is imposed include the following:

1. To flash the ECU with new software
2. To run specific diagnostic routines
3. To read specific memory locations in EEPROM
4. To read certain data from EEPROM
5. To write certain data into EEPROM
6. Clear certain Diagnostic Trouble Codes (DTCs) in EEPROM

Security threats caused by inadvertent data upload to ECU or improper routine run in ECU can lead to non compliance of the ECU to emission, safety and security standards. Additionally it can result in serious damage to ECU electronics and accessories. Vulnerability

of the ECU in such situations is avoided to a great extent by security access (0×27) service of ISO 14229. An overview of this service is shown in [Figure 1](#).

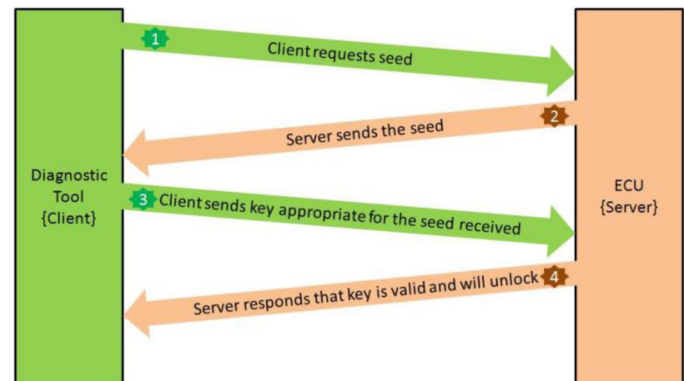


Figure 1. Overview of security access (0×27) service

A client (example: diagnostic tool) will request the server (example: ECU) to unlock it by sending the security access 'Request Seed' message. This message will also contain the security level required. A sample 'Request Seed' message is shown in [Table 1](#) where the server is in a locked state. This message asks the server to send the seed value to the client. Security level (in this case 0×01) is an OEM dependant parameter.

Table 1. Security Access - Request seed message example

Message source → Destination	Client → Server	
Message type	Request	
Byte Position	Content	Remarks
Byte 0	0x02	Single Frame, 2 bytes of data
Byte 1	0x27	Security Access Request Service ID
Byte 2	0x01	Security Access Type: Request Seed
Byte 3- Byte 7	0x00	Miscellaneous parameters

The server will respond by sending a positive message along with the seed value of 2 bytes size (Seed value can be of any size). Byte 2 in both the messages will be the same. A sample positive response message for the request message in [Table 1](#) is given in [Table 2](#). The seed value received is 0x2546 in this sample.

Table 2. Security Access - Positive response message example

Message source → Destination	Server → Client	
Message type	Response	
Byte Position	Content	Remarks
Byte 0	0x04	Single Frame, 4 bytes of data
Byte 1	0x67	Security Access Response Service ID
Byte 2	0x01	Security Access Type: Request Seed
Byte 3	0x25	Security Seed – High Byte
Byte 4	0x46	Security Seed – Low Byte
Byte 5- Byte 7	00	Miscellaneous parameters

There are two kinds of seed values, static and dynamic. When the client requests a seed from the server using 0x27 diagnostic service, the same seed value will be sent out each time in static seed, and different seed values in dynamic seed. The modern vehicles mostly use the dynamic seed system as it offers better security.

Based on the seed value received, the client can calculate key value “using a special algorithm” revealed only to the server and client. The Client can send the key to the server using the security access ‘Send Key’ message. An example of such a message is shown in [Table 3](#). The Security access type in [Table 3](#) is the security access type in [Table 1](#) incremented by one.

For the sake of simplicity, the key generation algorithm used here was done by just adding 0x1111 to the seed value. In usual cases the algorithm used will be complex with multiple computations.

Table 3. Security Access - Send Key message example

Message Direction	Client → Server	
Message type	Request	
Byte Position	Content	Remarks
Byte 0	0x04	Single Frame, 4 bytes of data
Byte 1	0x27	Security Access Request Service ID
Byte 2	0x02	Security Access Type: Send Key
Byte 3	0x36	Security Key – High Byte
Byte 4	0x57	Security Key – Low Byte
Byte 5 - Byte 7	00	Miscellaneous parameters

The Server can also calculate the key value for the generated seed. If the key value received from the client is matching with the server generated key, the ECU will respond to the client through a positive response message that it is unlocked. An example of such a message is shown in [Table 4](#). Once the ECU is unlocked, the client can read data, write data, and run diagnostic routines within the current security level. Different OEMs have different requirements for keeping the ECU in an unlocked state. One example may be sending the tester present message (0x3E) cyclically with an appropriate time interval. This is the security access service explained in the latest version of ISO14229-1:2013

Table 4. Security Access - ECU unlocked message example

Message source → Destination	Server → Client	
Message type	Response	
Byte Position	Content	Remarks
Byte 0	02	Single Frame, 2 Bytes of data
Byte 1	0x67	Security Access Response Service ID
Byte 2	0x02	Security Access Type: Send Key
Byte 3 - Byte 7	00	Miscellaneous parameters

It is evident from the message traces that the means to unlock the ECU is the algorithm used for calculating the key from a seed. It is evident that the ‘password is algorithm itself’. There exist a few security holes in this system which may allow intruders to unlock the ECU even if they are unaware of the key generation algorithm.

Modern vehicles are highly sophisticated with an abundant feature list. Apart from luxury and comfort, customers are increasingly looking at the safety and security of vehicles. ECU security becomes important in this perspective. If intruders can unlock the ECU, and tamper with the data, it can lead to hazardous conditions and jeopardize the safety of its passengers. Current advances in information security and encryption technology could be adopted in the automotive domain also.

There is a large amount of information available on threats and vulnerabilities on vehicle security. [1] Explains the basic hardware, software and their arrangement required for intruding into a CAN bus. It lists down commercial sites to purchase emulation tools. [2] Explains the basics of UDS and CAN from an intruder's perspective. [3] Makes a comprehensive experimental analysis of security attacks on ECUs. [4] Explains security loopholes in a vehicle leading to such hazardous situations like disabling of brakes, selectively braking individual wheels on demand, stopping the engine, and so on. [5] Explains about a reported case where OBD port and PC based tools were used to make the car jerk left and right, controlling the horn, triggering the seat belt tensioner and even entirely turning off the brakes. Cases of getting control of the CAN bus and thereby cluster, doors etc are also reported. It can be concluded that the most significant automotive interface, OBD-II port, which is federally mandated in the U.S. giving direct access to automobile's key CAN buses, can provide sufficient access to compromise the full range of automotive electronic systems. This paper is aimed at pointing out the short comings of security access (0x27) feature of UDS along with OEM specific algorithms, accessed through OBD port, and proposing a new security access system.

Encryption method proposed in this paper can be used as an add-on to the latest ISO 14229 standard. The paper is organized into 2 sections - Section 'Issues in the existing system' explains different ways to unlock the ECU without knowing the key generation algorithm and section 'ADvanced Encryption STandard (ADESTA) for Diagnostics over CAN' explains an Advanced Encryption Standard capable of nullifying those issues. Message traces are used to demonstrate the concepts. This paper brings out the security flaws in the existing systems and puts forth a proposition for a better solution. Since UDS security access service has been widely accepted, and prevalent for long, any new security method is bound to build upon its strengths.

ISSUES IN THE EXISTING SYSTEM

Security flaws in the UDS security access service (0x27) are explained in this section. A multitude of hardware devices and software applications are available in the market today which can emulate CAN messages and also provide the option to view them in a desired format when they are connected to the vehicle through the OBD port. Some of such devices come with bundled software also. Any such device would be required to prove the security flaws present in the existing system. An automotive expert can identify CAN ID of the diagnostic messages from the message traces. Diagnostic CAN messages with this CAN ID will be sent from the emulation tool to the ECU to demonstrate the security flaws.

Earlier such access through the OBD port was achieved using dedicated handheld tools like Ford's NGS, Nissan's Consult II and Toyota's Diagnostic Tester which were programmed via Windows based computers. Nowadays such access is realized mainly by a separate hardware unit which connects the OBD port to the computer through USB /Wi-Fi interface. Software residing in the computer can analyze and emulate CAN messages. Some such devices are Toyota's TIS, Ford's VCM, Nissan's Consult 3 and Honda's HDS.

Security Hole 1 - For Static Seed

As the Seed value would be fixed in the case of the static seed system, it is more vulnerable to threats. Whenever a client requests a seed from a server using 0x27 0x01 hex diagnostic service, server or the ECU will always reply with the same seed value. A message trace of consecutive seed requests and corresponding responses are shown in [Table 5](#). It is evident that the multiple seed requests from the client always return the same seed value.

Table 5. Message trace for 27 01 - Static seed system

Time (ms)	Message Type	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5-7
1	Request	0x02	0x27	0x01	00	00	00
2	Response	0x04	0x67	0x01	0x22	0x55	00
3	Request	0x02	0x27	0x01	00	00	00
4	Response	0x04	0x67	0x01	0x22	0x55	00
5	Request	0x02	0x27	0x01	00	00	00
6	Response	0x04	0x67	0x01	0x22	0x55	00

In most of the cases the length of the seed and the key shall be the same, let this be 'n' bits. If the length of seed is 2 bytes then the length of the key will also be 2 bytes. In the above example value of 'n' is 16 bits. The algorithm which generates 2 byte key from a 2 byte seed can be decoded through a trial and error mechanism, though it will consume some time.

If n=16 bits, the correct key will be one among the 2^{16} combinations possible. All these 2^{16} combinations can be tried to see when the server gives a positive response. A possible scenario where one could crack a static seed system is explained in [Table 6](#). The steps could be automated with a piece of software.

Table 6. Sample scenario - security hole 1

Step 1	Set i=1.
Step 2	Send a 0x27 0x01 from Emulation tool to ECU.
Step 3	Analyze the nonzero bytes in the response message after 0x67 0x01 and thus find the value of n.
Step 4	Create a table B of 2^n different possible combinations for the key value.
Step 5	Send the i^{th} key value from the table B to ECU using 0x27 0x02.
Step 6	If ECU is not unlocked due to wrong key value, set $i=i+1$, and execute step 2, step 5 and step 6 in sequential order.