Securing the Connected Car with Hardware Based Security

Rajeev Gulati

Data I/O Corporation

6645 185th Ave NE, Suite 100
Redmond, WA USA 98052

+1 (425) 881-6444

Advancements in the connected and autonomous vehicles are driving new architectural designs to ECUs that connect various subsystems together and expose them to the outside world over various networks. These advancements increase the risk of threat actors gaining access to the subsystems. This paper examines foundational security requirements for a hardware based approach to secure the connected car and methods to inject cryptographic elements into the ECUs in production.

11/21/2017

Introduction

The automotive market is being disrupted today by three significant trends- **electrification of the car**, **autonomous driving** and **Connectivity**. The electrification of the car is leading to the replacement of mechanical systems in a car by electronic systems, resulting in highly computerized cars. Autonomous driving requires not only more computational horse power in a car to service events in real time, but also the exchange of data between sub-systems within a car and outside a car (V2V, V2I). Connectivity to the Internet allows diagnostic and customer relevant data generated in the car to be shared with the outside world and intelligent services to be delivered to the car over wireless networks.

While the confluence of these trends will create an autonomous vehicle with much enhanced experiences for the driver and passengers of the car, the pervasive use of electronics and computers (e.g. ECU's, Gateways, modem chips) for processing and communications within and outside the vehicle exponentially increases security and safety concerns within a car due to the introduction of numerous new security vulnerabilities and threat vectors.

Security Issues and Implications to Connected Car Architecture

A Typical connected car architecture has a number of engine control units (ECUs) that perform various functions within a car (e.g. Telematics, Steering, Braking, Transmission, Lighting, ADAS, TPMS, Entertainment). These subsystems in the car are generally connected to each other and to the outside world over a network (e.g. Can Bus, Ethernet, WiFi).

If a threat actor can get access to an external communication interface into a car (e.g. the telematics control unit), then he/she can send spurious messages to other systems in the car (steering, braking) and compromise secure and safe operation of the car. Thus it is important to ensure that **external interfaces** into the car be **secured**. This can be achieved by ensuring that only certain "known and eligible" entities can communicate with the car from the outside world.

Most car architectures today are also monolithic in nature where there is no segregation of external interfaces from the internal subsystems in the car. Additionally, there is very little separation of the various functional sub-systems inside a car- if a threat actor can get access to one subsystem of the car, it may also be able to take over another subsystem. To overcome this architectural limitation and **enhance security** of the car, it is important to create **multiple functional domains** within a car that are separate from each other but networked through a "gateway ECU" subsystem. The gateway ECU is also the pathway to any subsystems that are connected from inside the car (e.g. telematics unit) to the outside world.

There are multiple advantages in relation to security provided by a **multi-domain** architecture. One advantage is that the **gateway** provides **functional isolation** within a car, so if one subset of the car is attacked, other systems can efficiently be isolated and protected. The **gateway** also **monitors communication** between various ECUs in a car and ensures that messages that are sent and received within the car subsystems **are authentic and have not been modified by a bad actor**. The **gateway** can also **firewall** the **external network** (and subsystems connected to it) from the internal network of the car. With modern technology, the latencies introduced by such gateway functions can be easily managed in the design of the system. In summary, introducing **domain separation** and **authentication** for domain access in the architecture makes the car **more secure** because it **reduces the surface area of attack** for bad actors.

The next area of **vulnerability** in the car is the **network**. A bad actor who can get access to the network within the car can manipulate data (or steal data) that is being exchanged between ECUs, which could lead to unsafe and insecure operation. A secure network in the car needs to ensure that data exchange between different ECU is protected- **source and destination of data** is **authenticated, data is encrypted during communication and protected from modification by bad actors**.

Another area of **vulnerability** in the car is the **domain specific ECU**, that executes the firmware and processes data to support various features of a connected car. To prevent firmware from getting modified by a bad actor, it is important that the **firmware running on the ECU be stored in a locked storage area**. In addition, prior to executing the firmware on the ECU, it is important that the ECU **perform a check on the firmware image to ensure it is from a trusted source and has not been altered since it was stored**. The ECU also needs to ensure that **when FW is running on the ECU, it cannot be intercepted by bad actors** so that they can change the behavior and functionality of the firmware running on the ECU. In addition, **ECUs should be able to authenticate the source of the communicated data, encrypt/decrypt outgoing/incoming data and validate that received data has not been modified in transit**. Given rapid software and firmware development cycles, it is not uncommon for firmware to have operational bugs or security vulnerabilities in the field. Thus it is also important for **ECUs to be able to update their firmware securely during field operation**.

In summary, there are four key areas in a connected car that need to be secured and protected- the **external interfaces**, **the gateway**, **the network** and the **ECU**.

Foundational Security Requirements for the Connected Car

In order to secure the four key areas in the car described above, various components of the connected car architecture have to embody certain security features. These features are also known as **Foundational Roots-of-Trust** (RoTs)

The first requirement is that each computational device (e.g. ECU, gateway) have a **Unique Identity** that has been embedded in the device in a trusted way. In addition to Unique Identity, an additional requirement is that each computational device in the car be able to **mutually authenticate itself** with other devices (ECUs) on the car network. This is how devices can ensure if they are communicating with the correct devices both inside and outside the car, thus protecting the external interfaces.

In addition to Identity and Authenticity, each ECU should have ability to **Sign Data** or **Check Signatures** on Data. This is done through a cryptographic process of "Data Signing" or "Signature Verification". This helps each ECU determine the **authenticity of the source of the data** (that it receives) and helps ensure the **data is not modified** by a bad actor on transit. The same process is also used to ensure **authenticity** and **sanctity** of firmware before it is **executed** on the ECU. These requirements help **ECUs boot securely and only execute authenticated code.**

To protect the confidentiality of data, each ECU in the car should also have capability to **encrypt/decrypt** outgoing**/**incoming data.

The embodiment of foundational roots of trusts (Identity, Authenticity, Data and Code Signing, Encryption and Decryption) on ECU's requires two additional and important features on ECU's. **Secure nonvolatile storage (Fuses, OTP storage, Flash Memory)** is required so that data (private and public keys and cryptographic secrets) and firmware that is associated with these cryptographic operations can be stored securely so that it cannot be tampered or modified. The ECU's should also provide a **Secure Execution Environment** such that the firmware that executes (and the data it manipulates) on the ECU in an environment where it is protected from external bad actors (like malware or data intrusion).

Cryptographic operations required to support the various foundational roots of trust are computationally intensive and should be supported on the **ECU through hardware based cryptographic acceleration.**

In summary, each computational device in the car should embody the following foundational Roots of Trust:

1. Identity and Authenticity,
2. Ability to Encrypt/Decrypt Data,
3. Ability to Sign Data,
4. Ability to Verify Data,
5. Cryptographic Acceleration,

6. Secure Storage and a
7. Secure Execution Environment.

All the features and functionality required to build and secure a connected car (including a firmware update service) can be built using these foundations to secure four key areas of the car described earlier.

Servicing the Security Requirements via HW vs SW

Now that we understand the foundational security requirements of the Connected Car, let's look at options to best service these requirements.

When thinking about servicing the security requirements in a car, we should apply lessons that we have learned from the PC, Server, Cloud, Smartcard and the Mobile phone industry. In each of these cases, **foundational security requirements are met through hardware based mechanisms as opposed to software based mechanisms**. Industry and security experts agree that **HW based mechanisms**, once enabled in hardware, are superior to SW based mechanisms because they cannot be **mutated, contaminated** or **infected** by external bad actors. Enabling Security as early as possible in a electronic device life cycle (as early as manufacturing of the electronic device) is much preferred from a security perspective and HW based security can be easily enabled in manufacturing in a scalable, reliable and cost effective way.

When thinking about servicing the security requirements for ECUs through HW, there are **two options** that are available to car OEMs and their tier-1 manufacturers.

The **first option** is to add security features to an ECU based subsystem by coupling a traditional MCU with a Secure Element (SE) chip on the ECU PCB. In this option, the ECU relies on the Secure Element to provide all the foundational security requirements of ECU-Identity, Authenticate, Ability to Encrypt/Decrypt Data,

Ability to Sign Data, Ability to Verify Data, Cryptographic Acceleration, a Secure Storage and a Secure Execution Environment.

Secure Elements are readily available from Silicon Vendors like NXP, Infineon, Microchip, Maxim, ST etc. This is indeed the strategy that is deployed in PC's, servers in the cloud, smart cards and mobile phones.

The **second option** is to use a Secure-MCU (instead of general purpose MCU) that integrates a Secure Element (or its feature set) into the MCU for a single chip solution. This is a new architectural direction for ECUs with number of solutions soon to the market like Synergy products from Renesas, PSOC 6 from Cypress and Kinetis from Freescale.

In summary, in order to secure a connected car, car manufacturers **need to ensure that the ECU based subsystems are enabled with HW based security** by leveraging either option described above, otherwise the connected car cannot be adequately secured.

Provisioning Security into SE's and S-MCU's

Designing ECU based subsystems with proper HW based mechanisms is a necessary first step towards securing the car. The next step is to **provision** (initially enable) the **security features** that are supported by the subsystems of the car. This is accomplished during **the early phases of manufacture of ECU based subsystems** using **device programming**.

Current methods to secure data and firmware are typically performed at the end of the manufacturing line and separated from the data and firmware programming process. Provisioning S-MCUs late in the manufacturing process increases the threat surfacearea and exposes security risks to the device including unauthenticated S-MCUs used in manufacturing, injection of malware and loss of IP.

During **device pre-programming, component devices like SEs and S-MCUs are securely embedded and enabled with the foundational roots-of-trust including:**

1. Cryptographic keys to establish identity and authenticity
2. Cryptographic keys to enable encryption/decryption
3. Certificates for Public Identity
4. FW is programmed into secure NVM of S-MCU
5. Secure execution environment is set up on S-MCU
6. Secure boot is enabled on S-MCU,
    a. SE or S-MCU is locked to immutable state
    b. subsystem enabled for secure FW update.

After the device is securely provisioned, the device now supports all the foundational security features that are the building blocks for making the various areas of the car secure.

Role of Data I/O:

Data I/O is an industry leader in providing device programming systems to the Automotive Electronic manufacturing industry. Data I/O PSV data programming systems are used at tier 1 OEM factories, Programming Centers and Contract Manufactures worldwide, to bring smart devices to life, by programming firmware and data into flash memory, flash based microcontrollers and secure elements.

In order to service new requirements related to "Security Provisioning," Data I/O has created the **SentriX™** (www.dataio.com/SentriX) Security Provisioning System. This system is used for "Security Provisioning" of devices like Secure Elements and Security Microcontrollers so that OEMs can embed Security into devices in manufacturing as an incremental process (incremental to firmware/data programming) in a scalable and cost effective way.

An important benefit of the SentriX system is that **it enables OEMs to secure the Supply Chain**. Before provisioning devices, the SentriX system can **check if the programmable component came from an authentic vendor** by authenticating the existing security credentials on a programmable device. As part of the security provisioning process, the SentriX system **creates device identities that can be checked for validity** by later phases of manufacturing process (e.g. system test). The SentriX system also **provides control to OEMs to manage the quantity of devices produced** by a programming center or a contract manufacturer. It also helps **OEMs protect their FW** from **modification or theft** by protecting it from point of creation (OEM lab) to point of programming/manufacture (Programming Center or Contract Manufacturer).

Data I/O is working closely with SE Silicon Vendors, S-MCU Vendors and programming center partners to enable the Security provisioning of SE's and S-MCU in manufacturing.

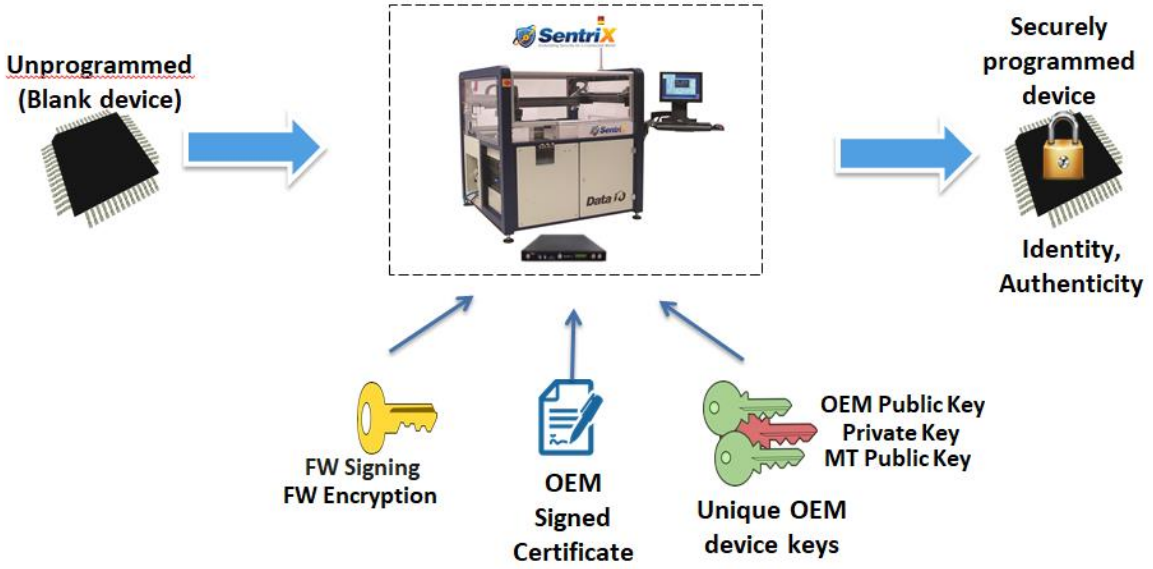
Tier-1 OEM's who build ECU based subsystems for the car can deploy Data I/O SentriX systems directly in their manufacturing factories or leverage programming and provisioning services offered by logistics partners (Avnet, Arrow etc.) to secure their ECU based subsystems.

Summary

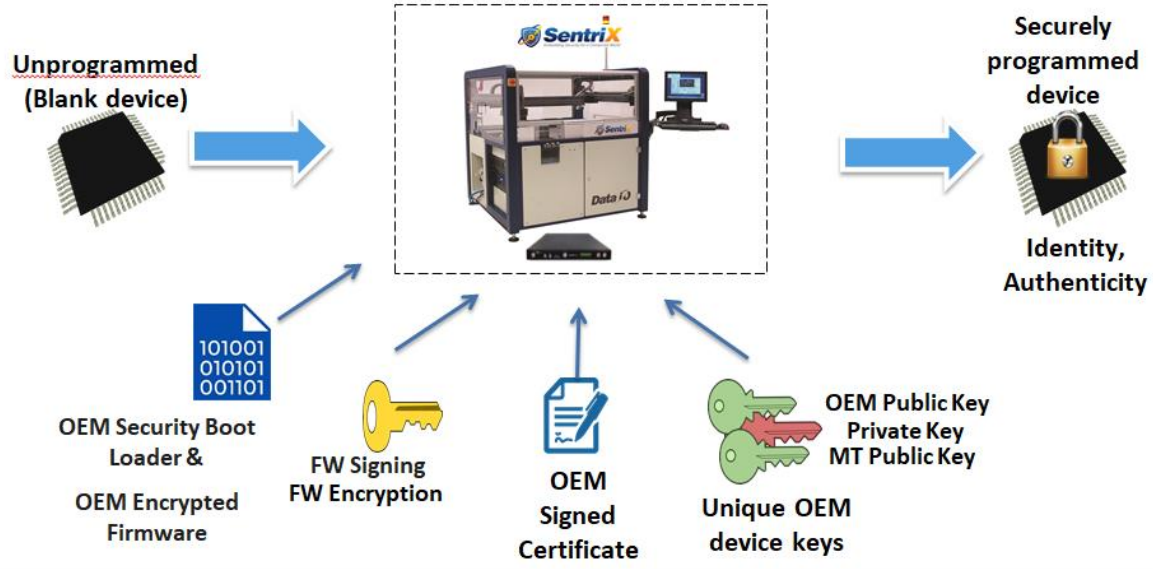
Leading automotive manufacturers are demanding ever greater security for their products throughout their supply chain. Establishing true hardware based security on ECUs is now possible by integrating Secure elements or Secure MCUs into these designs. Data I/O is enabling this market by providing secure provisioning systems for the manufacturing environment, and working closely with leading Silicon suppliers. Tools suppliers, and downstream services suppliers to simplify the process.

Data I/O Corporation
6645 185th Avenue NE, Suite 100 Redmond, WA 98052 USA
www.dataio.com
+1 (425) 881-6444

Secure Element Provisioning Using the SentriX platform



Secure MCU Provisioning Using the SentriX platform



Data I/O Corporation
6645 185th Avenue NE, Suite 100 Redmond, WA 98052 USA
www.dataio.com
+1 (425) 881-6444