Formally Verifiable Features in Embedded Vehicular Security Systems

Gyesik Lee
(ROSAEC Center, Seoul National University, Korea)

Hisashi Oguma, Akira Yoshioka
(Toyota Infotechnology Center, Japan)

Rie Shigetomi, Akira Otsuka, Hideki Imai
(RCIS, AIST, Japan)

VNC 2009, Tokyo

Background: Safety application with V2V/R2V communication

- Troubles with Electronic Control Unit (ECU)
  - Downloading malicious programs
  - Tampering with genuine ECU programs
  - Paralizing transportation systems with malicious information
- Our attention
  - Remote Attestation Scheme: Execution of trusted programs (Trusted booting)
  - Secure and authenticated communication: Encrypted or signed messages with a lightweight encryption system

Is this enougn?

Where is the guarantee that the protocols satisfy the expected behavior or security properties?

Background: Embedding security in vehicles

Wolf et al. (2007), "State of the Art: Embedding Security in Vehicles"

(P1) Only valid controllers can communicate.

(P2) All unauthorized messages are to be processed separately or immediately discarded.

(P3) Every communication is based on encryption and authentication in order to provide confidentiality and authenticity of exchanged data.

(P4) A single successful attack should not endanger the whole system.

Background: Embedding security in vehicles

Wolf et al. (2007), "State of the Art: Embedding Security in Vehicles"

(P1) Only valid controllers can communicate.

(P2) All unauthorized messages are to be processed separately or immediately discarded.

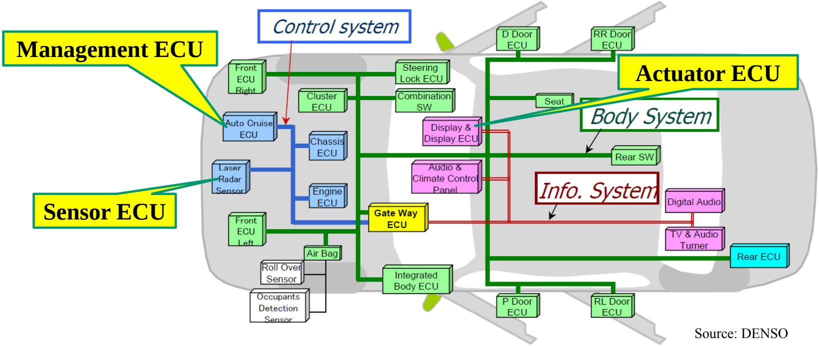
(P3) Every communication is based on encryption and authentication in order to provide confidentiality and authenticity of exchanged data.

(P4) A single successful attack should not endanger the whole system.

(P5) It is desirable that a software security module can be verified formally.

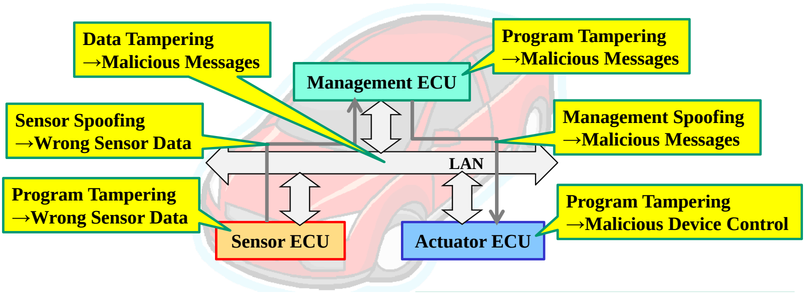
In-Vehicle Communication Architecture

- Assumption: All devices linked by LAN (CAN, LIN, or FlexRay)
- Each device is represented by an ECU
  - 3 categories: sensors, actuators, and management
  - All ECUs communicate with each other through LAN

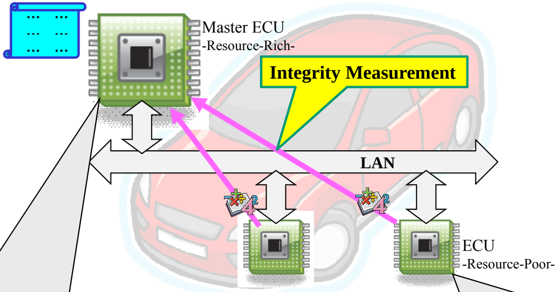


Source: DENSO

Threats for in-vehicle communication



Attestation-based scheme for in-vehicle communication

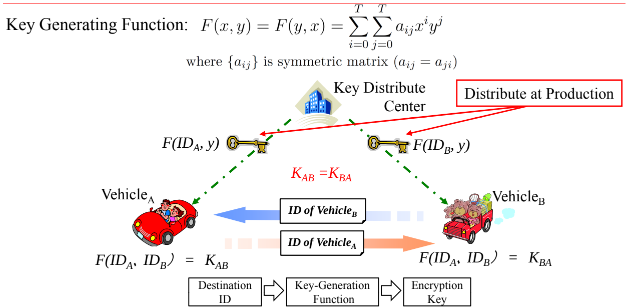


Master ECU
- White List: Genuine Hash Value of Each ECU
- Verify every ECUs with the White List
- Distribute "Attestation Token" for Verified ECUs
- Use the Token As a Part of the Encryption Key

Each ECU ($\neq$ Master ECU)
- Hash Own Code
- Send the Value to Master ECU
- Retrieve "Attestation Token"
- Use the Token in Talking with Others

A symmetric encryption scheme: KPS

- Basic requirements: low latency and long lifetime



- Security parameter: T
  - For extracting key-generation function
  - Required to tamper with $(T+1)$ ECUs

Initialization protocol

$$
\begin{array}{lll}
\text{(Step 1)}\ \mathcal{E}_m \rightarrow \text{all}: & & \text{broadcast } r_1 \\
\text{(Step 2)}\ \mathcal{E}_x \rightarrow \mathcal{E}_m: & & Sig_{F_x(\mathcal{E}_m)}\{\mathcal{E}_x, H(\mathrm{ROM_x}), r_1, r\} \\
\text{(Step 3)}\ \mathcal{E}_m \rightarrow \mathcal{E}_x: & & \{r_2, r, H(\mathrm{ROM_m})\}_{F_m(\mathcal{E}_x)}
\end{array}
$$

r_1, r_2: Random numbers generated by Master ECU
r: Random number independently generated by each ECU

RoSAEC TOYOTA InfoTechnology Center Co., Ltd. AIST

The protocol problems

- Good cryptography alone is not sufficient for writing good security protocols. Even with a perfect cryptography.
- In general, it is very hard to detect security flaws.

Photo by Cas Cremers:

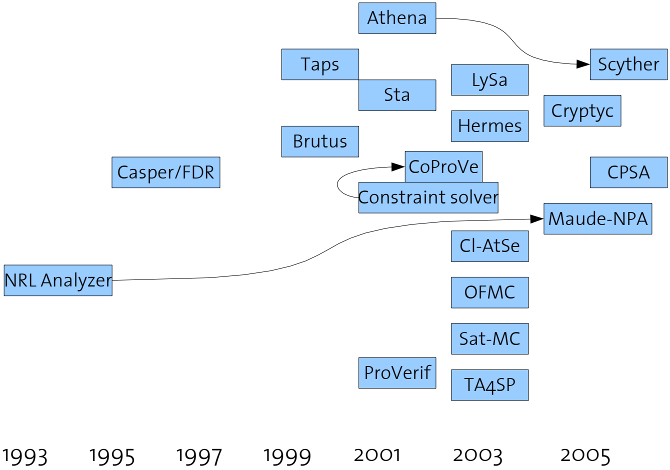
Automatic protocol verification I

- The design of protocols is error-prone.
- Errors cannot be detected by testing, since they appear only in the presence of a malicious adversary.
- Typically achieved using language-based techniques
- Verification of protocols in the Dolev-Yao model
- Unbounded number of sessions

Automatic protocol verification II

- Protocol insecurity is NP-complete for a bounded number of sessions.
- Undecidable for an unbounded number of sessions.
- Automatic verification for an unbounded number of sessions cannot be achieved for all protocols.

(Semi-)Automatic verification tools



(Cas Cremers)

ProVerif

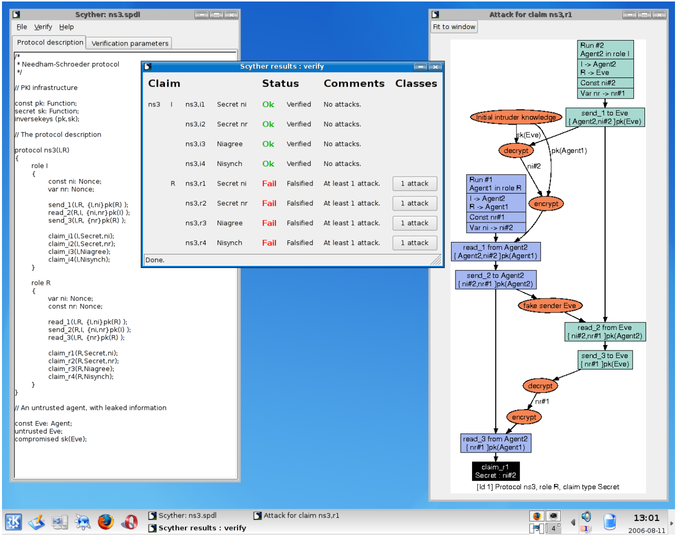
- An abstract representation of the protocol by a set of Horn clauses.
- Fully automatic proofs of protocols for an unbounded number of runs and an unbounded message space.

$$\frac{adversary(\{M\}_k) \qquad adversary(k)}{adversary(M)}$$

$$\frac{adversary(\langle M\rangle_{sk_A})}{adversary(M)} \qquad \frac{adversary(M) \qquad adversary(k)}{adversary(\langle M\rangle_k)}$$

$$\frac{}{adversary(pk(sk_A))}$$

Scyther


Scyther: ns3.spdl
File Verify Help
Protocol description
Verification parameters
/*
* Needham-Schroeder protocol
*/
// PKI infrastructure
const pk: Function;
secret sk: Function;
inversekeys (pk,sk);
// The protocol description
protocol ns3(I,R)
{
role I
{
const ni: Nonce;
var nr: Nonce;
send_1(I,R, {I,ni}pk(R));
read_2(R,I, {ni,nr}pk(I));
send_3(I,R, {nr}pk(R));
claim_i1(I,Secret,ni);
claim_i2(I,Secret,nr);
claim_i3(I,Niagree);
claim_i4(I,Nisynch);
}
role R
{
var ni: Nonce;
const nr: Nonce;
read_1(I,R, {I,ni}pk(R));
send_2(R,I, {ni,nr}pk(I));
read_3(I,R, {nr}pk(R));
claim_r1(R,Secret,ni);
claim_r2(R,Secret,nr);
claim_r3(R,Niagree);
claim_r4(R,Nisynch);
}
}
// An untrusted agent, with leaked information
const Eve: Agent;
untrusted Eve;
compromised sk(Eve);
Scyther results : verify
Claim Status Comments Classes
ns3 I ns3,i1 Secret ni Ok Verified No attacks.
ns3,i2 Secret nr Ok Verified No attacks.
ns3,i3 Niagree Ok Verified No attacks.
ns3,i4 Nisynch Ok Verified No attacks.
R ns3,r1 Secret ni Fail Falsified At least 1 attack. 1 attack
ns3,r2 Secret nr Fail Falsified At least 1 attack. 1 attack
ns3,r3 Niagree Fail Falsified At least 1 attack. 1 attack
ns3,r4 Nisynch Fail Falsified At least 1 attack. 1 attack
Done.
Attack for claim ns3,r1
Fit to window
Run #2
Agent2 in role I
I -> Agent2
R -> Eve
Const ni#2
Var nr -> nr#1
Initial intruder knowledge
send_1 to Eve
{ Agent2,ni#2 }pk(Eve)
sk(Eve)
decrypt
pk(Agent1)
ni#2
Run #1
Agent1 in role R
I -> Agent2
R -> Agent1
Const nr#1
Var ni -> ni#2
encrypt
read_1 from Agent2
{ Agent2,ni#2 }pk(Agent1)
send_2 to Agent2
{ ni#2,nr#1 }pk(Agent2)
fake sender Eve
read_2 from Eve
{ ni#2,nr#1 }pk(Agent2)
send_3 to Eve
{ nr#1 }pk(Eve)
decrypt
nr#1
encrypt
read_3 from Agent2
{ nr#1 }pk(Agent1)
claim_r1
Secret : ni#2
[Id 1] Protocol ns3, role R, claim type Secret
Scyther: ns3.spdl
Attack for claim ns3,r1
Scyther results : verify
13:01
2006-08-11

Some features of ProVerif and Scyther

- fully automatic
- very fast: small examples verified in 0.x s; complex ones in few minutes
- very precise in tests for secrecy and authentication
- for unbounded number of sessions and message space
- applicable for a wide range of cryptographic primitives

Conclusion

- Attestation-based security scheme for in-vehicle communication
- Important basic features in embedding security in vehicles.
- Introduction to formal methods in (semi-)automatic verification of security protocols.
- Application in in-vehicle communication.
- Time for more interest in using formal methods!