

Formally Verifiable Features in Embedded Vehicular Security Systems

Gyesik Lee, Hisashi Oguma, Akira Yoshioka, Rie Shigetomi, Akira Otsuka and Hideki Imai

Abstract—In an overview paper called *State of the Art: Embedding Security in Vehicles*, Wolf et al. give a general state-of-the-art overview of IT security in vehicles and describe core security technologies and relevant security mechanisms. In this paper we show that a formal analysis of many of the related properties is possible. This indicates that many expected aspects in the design of vehicular security can be verified formally. Our presentation is based on a recent paper by the second author et al. [13] where a new attestation-based vehicular security systems is represented. We briefly summarize the general properties required in the design of vehicular IT security and verify that the new architecture given by Oguma et al. suggests new desirable security aspects.

I. INTRODUCTION

In the past four decades, the number and sophistication of electronic systems in vehicles have greatly increased. Today, the cost of the electronics in commonly-used vehicles can amount to more than 23 percent of the total manufacturing cost. Analysts estimate that more than 80 percent of all automotive innovations now stems from the research on electronic devices, cf. [8].

Most vehicle services are implemented with such electronic devices, which are called electronic control units (ECUs). All of the ECUs in a vehicle can communicate with each other through a controller area network (CAN). Sophisticated services, such as electronic stability control and anti-lock braking system, are implemented with collaborating ECUs. Consequently, the size of the software in vehicles has increased and the dependency among the ECUs has become complicated.

Furthermore, the danger of facing harmful effects is also increasing which can be caused by software bugs. An example of this kind of problem is the engine stall trouble resulted from an issue with the software in ECUs, cf. [16]. As well as such kind of ECUs, the software of navigation systems also is prone to software bugs [14]. Future navigation system will work with AT gear shift control, suspension damping force control, or driver's brake operation assistance [1]. If a media from which a navigation system loads map information has malware, it may attack the navigation system by abusing

a software bug and vehicle systems such as brake operation may receive fake messages from tampered navigation system. Besides technical deficiency, we have to consider the issue of the network security protocols for in-vehicle systems.

Network security protocols are usually based on cryptographic primitives, and their analysis is one of the most challenging tasks because it involves many subfields such as cryptosystems, signature schemes, secure hash functions, transfer mechanisms, and secure multiparty function evaluation methods. Furthermore, it is vulnerable to intruders in the network who may have control of one or more network principals. Therefore, network protocols are often subject to non-intuitive attacks. A security protocol must be able to achieve its goals in face of these hostile intruders.

It seems nowadays inevitably required to verify that a security protocol satisfies its requirements based on a formal method which is based on a combination of a mathematical or logical model of a system and its requirements. Actually, the application of formal methods to cryptographic protocol analysis has been investigated since almost 30 years, cf. Meadow [11]. An important area is the development of tools for automatic verification of security protocols allowing unbounded number of sessions.

In this paper, we choose the attestation scheme presented in Oguma et al. [13] as an example of formal method verification using Blanchet's ProVerif [3]. We have assumed that vehicles have connecting capability to other vehicles, road, as well as the Internet to supply an extensive number of services. At conventional remote attestation scheme, it is necessary to communicate with a verification server for attestation. However, satisfying such a requirement is hard because of a possible limitation of the mobility capability, e.g., in tunnels or underground parking spaces. The purpose of our attestation scheme is keeping software of vehicle system healthy. In this scheme, we let several rich-asset ECUs be Masters. A Master ECU verifies other ECUs in the role of a verification server.

Keeping software healthy under our attestation scheme, we first have to verify whether our attestation scheme logically satisfies correctness or not. In this paper, we show the verification result of our attestation scheme based on a formal method.

The rest of this paper is organized as follows. Section II gives a brief explanation of the attestation based security architecture proposed in Oguma et al. [13]. Section III describes the KPS cryptographic basis used for the new security architecture. In Section IV, we give a brief summary of required features in the design of vehicular IT security, followed by a brief introduction to automatic protocol verifi-

For Gyesik Lee, this work was partially supported by the Engineering Research Center of Excellence Program of Korea Ministry of Education, Science and Technology(MEST) / Korea Science and Engineering Foundation(KOSEF), grant number R11-2008-007-01002-0.

G. Lee is with ROSAEC Center, Seoul National University, Korea gsllee@ropas.snu.ac.kr

H. Oguma and A. Yoshioka are with Toyota InfoTechnology Center, Co., Ltd, Japan {oguma, yoshioka}@jp.toyota-itc.com

R. Shigetomi, A. Otsuka and H. Imai are with Research Center for Information Security, National Institute of Advanced Industrial Science and Technology, Japan {rie-shigetomi, a-otsuka, h-imai}@aist.go.jp

cation in Section V. In Section VI, we give a detailed, formal analysis of the security protocol given in [13] and show that many security features can be verified formally.

II. RELATED WORK: ATTESTATION SCHEME FOR IN-VEHICLE SYSTEM

In a near future, the software architecture for vehicles will be standardized like PC/AT compatible machines[2]. On the other hand, technological innovations will enable vehicles to achieve both inter-vehicle and road-to-vehicle communications. Moreover, vehicles will be able to connect to the Internet to supply an extensive number of services. Given this situation, it is no exaggeration to claim that future vehicle systems will suffer from a wide variety of threats. If an ECU is in trouble because of a malicious software running on it, it might not work well and cause traffic disaster.

One plausible way to keep software healthy with which we can easily come up is adopting remote attestation scheme for vehicular security systems. Because, however, vehicles will then be equipped with mobility capability, we will be faced with a correspondingly hard task to keep stable connection with verification server and also to apply that scheme without any modification.

In [13], the second author et al. have proposed an attestation-based security scheme for in-vehicle communication as an alternative way of dealing with this issue. They assume that a novel vehicle will have many ECUs with poor asset and some with rich asset. They let rich-asset ECUs be Masters. A Master ECU verifies other ECUs in the role of a verification server. Furthermore, they adopt secret-key cryptography based on Key Predistribution System (KPS), a.k.a. Matsumoto-Imai scheme [10], and they let each pair of ECUs have a separate key. By adopting this scheme for communications between a server ECU and client ECUs, they have achieved both safety and low-latency communication between arbitrary ECUs under the scheme.

III. KPS AND AN ATTESTATION-BASED SECURITY

In practice, establishing a cryptographic system is usually done by a mixture of symmetric and asymmetric cryptography: a public-key scheme for key establishment and then a symmetric-key scheme for encrypted communications.

A symmetric-key encryption scheme provides a high security while storing secret keys for reuse is one of the main issues to be solved. Furthermore, key-management could be a difficult task when there are different keys for each pair of users. On the other hand, a public-key encryption scheme is useful for key establishment because of the simplicity in storing secret keys. The main disadvantage of using a public-key encryption scheme for vehicular embedded systems is its poor throughput performance because intensive arithmetic operations are necessary. Hence, it would not be so effective to use public-key encryption schemes for vehicular IT security where, besides safety and low-latency, low cost plays an important role because of a large amount of device.

A nice suggestion to solve this kind of dilemma was made by Matsumoto and Imai. In their seminal paper [10], the so-called Key Predistribution systems (KPS) are introduced. A

KPS consists of a KPS center and users who want to share a common key with the center. The KPS center possesses an algorithm which generates an individual secret algorithm for each user. These individual algorithms are predistributed by the center to the users and allow each user to calculate a common key from the ID of any other counterpart of a communication. That is, in KPS, no interaction between two participants is required in advance to share communication keys. Below is a formal definition.

A KPS consists of a center and multiple players P_i ($i = 0, 1, 2, \dots$). Given a prime number q and a security parameter T , the management center (i.e., vehicle manufacturer) randomly generates an $(T+1) \times (T+1)$ symmetric matrix $A = (a_{ij})$, i.e., $a_{ij} = a_{ji}$, and defines a bi-variate polynomial,

$$F(x, y) = \sum_{i,j=0}^T a_{ij} x^i y^j \pmod{q}.$$

The center delivers a key generation algorithm, $F_i(x) = F(x, i)$ to each player P_i . A shared key between any two players P_i and P_j is then $F_i(j)$ (resp. $F_j(i)$) generated by P_i (resp. P_j). It provides a symmetric-key encryption scheme because $F_i(j) = F_j(i)$ holds.

In a KPS, only privileged users can compute the decryption key for the encrypted data, and no other users can obtain any information about the key. This will be proved formally in Section VI. We refer to [12], [7], and [15] for an extensive reference list about KPS.

The fact that KPS can be a good alternative for the design of a vehicular security is well demonstrated in Oguma et al. [13] where a KPS for communication between a Master ECU and many other ECUs is established instead of using a public-key encryption scheme. For a tamper-free communication between ECUs, an attestation-based security architecture is proposed instead of adopting TPM.

A vehicle manufacturer (resp. a Master ECU) plays the center role while each ECU (resp. each poor-asset ECU) in a vehicle plays a user role. For each pair of ECUs a separate key is used. A Master ECU executes attestation process to verify the authenticity of other ECUs. That is, in addition to key-possession check, there is also a test for authorization process by checking whether there is a software hash-code in a list of valid hash-codes. Some basic experiments in [13] showed that KPS is much faster than public-key encryption schemes such as RSA or even elliptic curve cryptography. In this way, one could solve the problems which could arise with an implementation with TPM, i.e. cost-sensitivity rendered by many embedded ECUs with low capacity, low latency, and communication in exceptional cases.

IV. EMBEDDED VEHICULAR SECURITY

Combining cost-sensitivity with diverse computational capabilities of ECUs is a main hurdle to be overcome in the design of embedded devices. High security, in general, requires large memory and high performance of calculation. But adding complex security solutions could be too expensive because of high quantity of ECUs. Hence the computational

ability, and the memory-capacity of processors and their production cost are the main factors to be considered in an efficient way. This explains why the vehicular IT basically belongs to the field of embedded security.

Guaranteeing security and privacy of in-vehicle communications is a difficult task because it must embed security features in stringent real-time protocols for time-constrained vehicle systems. In particular, one expects more efficiency in time and procedure as well as in production cost. It is also different from conventional policies for computers. Storing all the codes and processing units in a tamper-free device is too costly, and such a device is not adequate for updating software which is a critical factor for security implementation. Both for flexible maintenance and safe code execution it is more feasible to apply remote attestation like Trusted Platform Module (TPM) of the Trusted Computing Group (TCG), cf. [17]. A verification server, located away from the target platform, collects and checks the measurement results. However, TPM is not so feasible for vehicular IT security although it provides high level security. This is because most ECUs are equipped with few resources and vehicle systems generally require low latency,¹ and there are situations where vehicles can not communicate with the verification server for attestation schemes.

There are three kinds of communication where vehicular IT security should be applied: in-vehicle communication, vehicle-to-vehicle communication, and vehicle-to-infrastructure communication. In Wolf et al. [18] a list of some points required in common for secure communication is mentioned:

- (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.

It is one of the main goals of this paper to show that a formal verification of the four features (P1) ~ (P4) is not just desirable, but can be realized using an existing tool for fully automatic verification of security protocols. Here we use ProVerif [3], but any other tool with corresponding capability will do the same thing.

V. AUTOMATIC VERIFICATION OF SECURITY PROTOCOLS

Automatic or semi-automatic protocol verification for bounded or unbounded number of sessions has become the main object of formal analysis of security protocols. In case of unbounded number of sessions, it is typically based on language-based techniques such as typing or abstract interpretation. There are many (semi-) automatic tools for

protocol verification. We chose for ProVerif to give a formal verification of the basic security features mentioned in Section IV.

ProVerif [3] is a leading automatic cryptographic protocol verifier in the so-called Dolev-Yao model [5]. Protocols are written by Horn clauses, and ProVerif checks full automatically whether some Horn clauses are derivable from the Horn clauses representing the protocol. It is sound in the sense that the security properties that it proves are really true. But there could be false attacks because of some approximations. Approximations occur during the translation of protocols written in the applied pi-calculus into Horn clauses. The idea is based on a simple correspondence between traces of communications and exchanges of information: received messages as the antecedents and sent message as the conclusion of a Horn clause.

ProVerif can handle many cryptographic primitives like shared- and public-key cryptography (encryption and signatures), hash functions, and Diffie-Hellman key agreements. Thanks to some approximations, it can handle an unbounded number of sessions of the protocol and an unbounded message space and can prove secrecy, authentication, strong secrecy, equivalences between processes that differ only by terms.

We refer to Blanchet [3] for a good comprehensive introduction into ProVerif. We also refer to Fournet and Abadi [6] which gives the analysis of a protocol for private authentication in the applied pi-calculus. A brief overview of other verifiers can be found e.g. in Cremers [4].

VI. A FORMAL VERIFICATION OF SECURITY FEATURES

A. Attestation protocol

Each ECU should be able to check the other ECU's identities and the genuineness of the received messages, i.e. whether the communication process is carried out by genuine ECUs. This is indeed one of the most basic and crucial points in designing a security architecture as mentioned in Oguma et al. [13]: (a) Each ECU communicates only with ECUs whose software configurations have been certified by the manufacturer. (b) In all communications between any two ECUs, the authenticity of each participating ECU and the integrity of each (possibly encrypted) message should be checkable. (c) All of this should work with replaced ECUs and remotely updated software.

In this section, we give a formal analysis of the security architecture in [13] and show thereby that all of the aforementioned requirements can be verified formally by ProVerif.

The proposed architecture consists of a center \mathcal{C} , a Master ECU $\mathcal{E}_0(= \mathcal{E}_m)$, and N -many ECUs \mathcal{E}_x , $x \in \{1, \dots, N\}$. In the rest of this paper x ranges over natural numbers up to N . During the production of a vehicle, each ECU is initialized by the center which possesses the pairs of a serial number and a key of each ECU: The Master ECU \mathcal{E}_m , e.g., gets and stores a KPS key-generation algorithm $F_0(\cdot)$ and a list of acceptable hash values for a type of vehicle where this Master ECU is implemented, $(version, H(\text{ROM}_x))_x$. Each

¹Low latency allows human-unnoticeable delays between an input being processed and the corresponding output providing real time characteristics.

ECU \mathcal{E}_x gets and stores the hash value $H(\text{ROM}_0)$ of the Master ECU and the key-generation algorithm $F_x(\cdot)$.

Every time when a vehicle is started, all the ECUs will be initialized by the Master ECU as follows: $\text{Sig}_k\{m\}$ (resp. $\{m\}_k$) denotes the signed (resp. encrypted) message m with a symmetric key k .

- (i) $\mathcal{E}_m \rightarrow \text{all}$: broadcast of r_1
- (ii) $\mathcal{E}_x \rightarrow \mathcal{E}_m$: $\text{Sig}_{F_x(0)}\{\mathcal{E}_x, H(\text{ROM}_x), r_1, r\}$
- (iii) $\mathcal{E}_m \rightarrow \mathcal{E}_x$: $\{r_2, r, H(\text{ROM}_m)\}_{F_0(x)}$

TABLE I
INITIALIZATION ATTESTATION

(i) The Master ECU \mathcal{E}_m generates random numbers r_1, r_2 and broadcasts r_1 to all ECUs. (ii) Each ECU \mathcal{E}_x runs an attestation process to the Master ECU by sending a signed message of the fresh hash value of its ROM $H(\text{ROM}_x)$ together with its serial number and a new nonce r (as a challenge to the following attestation protocol run by the Master ECU). (iii) If the signature check using $F_0(x)$ is successful and if the newly measured hash value $H(\text{ROM}_x)$ is in the list of valid hash values, the Master ECU measures its hash value and sends it back to \mathcal{E}_x encrypted together with r_2, r using the key $F_0(x)$.

In the protocol above there is a typical challenge-response communication between the Master ECU and each ECU using random numbers r_1, r_2, r , individual properties like hash values, and keys for cryptographic primitives. This kind of challenge-response communication provides entity authentication, i.e., one communication party identifies itself to a second party.

B. Automatic verification with ProVerif

Now we are going to analyze this protocol and verifies that it fulfills the security features mentioned in Section IV. Below is a ProVerif protocol for the initialization at start-up, written in the language of the applied pi-calculus: a Master ECU and two ECU \mathcal{E}_x and \mathcal{E}_c . \mathcal{E}_x represents an honest agent while \mathcal{E}_c a colluded agent. (Texts between $(*$ and $*)$ are comments and will be ignored by ProVerif.)²

a) *Preamble*: The preamble contains basic information about functions and their properties like for symmetric key cryptography, signature, KPS, hash function, etc. The information given here is, however, not just information, because the protocol behavior changes depending on it, hence could have impact on the security properties.

Declaring r_1 and r_2 in the preamble, for instance, means that they are global constants instead of being local. In this way, we can realize the broadcast of r_1 and the globalness of r_2 . r_1 and r_2 behave like a session identifier for each start-up.

We assume without loss of generality that there is only one public channel and no secret channels.

(* A public channel 'c'. *)

free c.

(*

Creation of r1 and r2 which act like a session identifier for each start-up.

r2 should be kept secret (hence 'private') if no ECU is colluded while r1 will be broadcasted publicly.

*)

free r1.

private free r2.

(*

Symmetric-key cryptography with two binary functions 'encrypt' and 'decrypt' and its decryption scheme.

*)

fun encrypt/2.

reduc decrypt(encrypt(x,y),y) = x.

(*

Signature function 'sign', 'checksign' for checking signer, and 'getmess' for getting message from a signed message without knowing who signed it.

*)

fun sign/2.

reduc checksign(sign(x,y),y) = x.

reduc getmess(sign(x,y)) = x.

(*

Key Predistribution System 'F' which corresponds to the KPS being used.

*)

private fun F/2.

equation F(x,y) = F(y,x).

(*

Hash function 'H' which could have been made public

*)

fun H/1.

(* ROM reading function 'ROM'. *)

fun Rom/1.

²The protocol can be copy-and-pasted into a ProVerif protocol.

b) *Queries*: ProVerif checks full automatically the validity of queries about secrecy of a message or about authentication of a communicating partner.

```
(*
  Queries about the secrecy of KPS and
  r2 with some compromised keys
*)
```

```
(* Query 1 *)
```

```
query attacker: F(hostX, hostM).
```

```
(* Query 2 *)
```

```
query attacker: r2.
```

The first two queries are about whether the attacker could get the key $F_x(0)$ or the secret session identifier r_2 . A negative answer for a query imply that secrecy is guaranteed. Since r_2 will be put into each outgoing message after the initialization, all messages without r_2 will be ignored.

The secrecy of $F_x(0)$ will not be endangered even in the presence of a colluded ECU \mathcal{E}_c . A compromised key does not endanger any other principal. In other words, to impersonate an uncorrupted ECU, just compromising an ECU is not enough to extract the key generation algorithm of another ECU. Moreover, the maximum collusion does not exceed the number of the implemented ECUs. This verifies (P4).

On the other hand, the secrecy of r_2 depends on the presence of a colluded ECU. If no ECU is colluded, then the session identifier r_2 remains secret. This implies that, without knowing the encryption key, one cannot perform the initialization and hence no communication as it is required in (P1) and (P2).

```
(*
  Queries about non-injective agreement
*)
```

```
(* Query 3 *)
```

```
query ev:e1(m,x,z,v) ==> ev:e3(m,x,z,v).
```

```
(* Query 4 *)
```

```
query ev:e2(m,y,z,v) ==> ev:e3(m,y,z,v).
```

```
(*
  Queries about injective agreement
*)
```

```
(* Query 5 *)
```

```
query evinj:eX(m,x,z,v,w)
  ==> evinj:eM(m,x,z,v,w).
```

The queries above are about authentication of each participating ECU. The first two correspond to the injective

agreement and the last one to non-injective agreement. (m stands for Master ECU \mathcal{E}_m , x for the honest ECU \mathcal{E}_x , and y for the compromised ECU \mathcal{E}_c .) See Lowe [9] for more about a hierarchy of authentication.

(Query 3) Before sending the second message r_2 , the master ECU can be sure that \mathcal{E}_x has previously been running the protocol, apparently with \mathcal{E}_m , and \mathcal{E}_x was acting as responder in his run, and the two agents agreed on the data values in $\{r_1, r\}$, and each such run of \mathcal{E}_m corresponds to a run of \mathcal{E}_x under the assumption that \mathcal{E}_x is not colluded.

(Query 5) A Positive answer to the query means that the protocol guarantees to the initiator, Master ECU \mathcal{E}_m , agreement with a responder, an ECU \mathcal{E}_x , on a set of data items $\{r_1, r, r_2\}$ in the following way: Whenever a ECU \mathcal{E}_x completes a run of the protocol, apparently with \mathcal{E}_m , then \mathcal{E}_m has previously been running the protocol, apparently with \mathcal{E}_x , and the two agents agreed on the data values in $\{r_1, r, r_2\}$, and each such run of \mathcal{E}_x corresponds to a *unique* run of \mathcal{E}_m . This means that, without knowing the symmetric key $F_x(0)$, the communication would fail. The injectivity means that the initialization process is totally secure and ensures authenticity of each ECUs, which corresponds to (P3).

On the other hand, the answer to (Query 4) is negative as expected. This is because the colluded ECU can send a message anytime. However, the positive answers to (Query 3) and (Query 5) indicate that it does not endanger the communication between two honest participants as required in (P4).

c) *Main part*: The main part describes the communication protocol in the language of the applied Pi-calculus. There are processes for each participant of the communication.

```
(*
  Protocol for the Master ECU
*)

let processM =

  (* (i) broadcast of r1 *)

  out(c, r1);

  (*
    (ii) reception of answers from hostB
  *)

  in(c,m);

  (*
    confirmation of the signature
  *)

  let (x_hostX, x_HRomX, =r1, x_r)
    = getmess(m) in
  if (x_hostX, x_HRomX, r1, x_r)
    = checksign(m, F(hostM, x_hostX)) then
```

```

(* attestation of hashed ROMs *)
if (x_hostX, x_HRomX)
  = (hostX, H(Rom(hostX))) then

(*
  authentication challenge of each ECU
*)

  event e1(hostM, hostX, r1, x_r)
else
  if (x_hostX, x_HRomX)
    = (hostC, H(Rom(hostC))) then
    event e2(hostM, hostC, r1, x_r);

(*
  (iii) answer to the challenge by
  sending r2
*)

event eM(hostM, x_hostX, r1, x_r, r2);
out(c, encrypt((r2, x_r, H(Rom(hostM))),
               F(hostM, x_hostX))).

```

Above is a description of the process from the viewpoint of the Master ECU while the following is that from the viewpoint of an ECU.

```

(*
  Protocol for each ECU
*)

let processX =

(* (i) reception of r1 *)

in(c, x_r1);

(*
  (ii) creation and sending of r with a
  signature
*)

new r;

(* answer to authentication challenge *)

event e3(hostM, hostX, x_r1, r);
out(c,
     sign((hostX, H(Rom(hostX)), x_r1, r),
          F(hostX, hostM)));

(*
  (iii) reception of the final answer
  from the Master ECU and attestation of
  the hashed ROM
*)

```

```

in(c, m);
let (x_r2, =r, x_HRomM)
  = decrypt(m, F(hostX, hostM)) in
if x_HRomM = H(Rom(hostM)) then

(*
  authentication challenge of the
  Master ECU
*)

event eX(hostM, hostX, x_r1, r, x_r2).

```

d) Protocol itself: One puts the processes for each participant together. First, one makes the name of the participants public and assumes some keys are compromised, i.e., the protocol will be checked that under the assumption that some symmetric keys are colluded. ProVerif checks then the queries under the assumption that the attacker could perform unlimited sessions of the protocol.

```

(*
  Protocol for start initialization
*)

process
  new hostM;
  new hostX;
  new hostC;

  (* making public the name of hosts *)

  out(c, hostM);
  out(c, hostX);
  out(c, hostC);

  (* hostC is compromised *)

  out(c, F(hostC, hostM));
  out(c, F(hostC, hostX));
  out(c, H(Rom(hostC)));

  (* unlimited number of sessions *)

  ((!processM) | (!processX))

```

If no ECU is compromised, all the answer will be as expected. That is, there is no attack for the secrecy of an uncompromised symmetric key or for the authentication process. Having finished the initial process, both sides could be sure of the authenticity of any other communicating partner.

After a successful initial attestation, each ECU keeps r_2 secret and places it in every communication with other ECUs as a proof that sender's identity is authenticated by the Master ECU. Furthermore, each ECU makes use of the KPS to create an encryption key for each communication, and each message - either signed or encrypted - contains a

counter. A counter can be used to prevent replay attacks.

VII. CONCLUSION

We have presented an approach to a formal verification of the features (P1) ~ (P4), mentioned in Section IV, which are required as basic for secure communication. In general, security facility has to ensure safety in itself and its logical soundness. The formal verification assures us that our attestation scheme for vehicle provides trust relationship and secure communication among all the ECUs. We strongly believe that our approach will not remain as a special case analysis in vehicular security. We will see more and more application of automatic verification tools.

REFERENCES

- [1] AISIN - investors - annual report -. Automotive parts and systems business. <http://www.aisin.com/finance/ir/report/02annual/pdf/auto.pdf#5>.
- [2] AUTOSAR. <http://www.autosar.org>.
- [3] Bruno Blanchet. Automatic verification of correspondences for security protocols. *Journal of Computer Security*, 17(4):363–434, 2009.
- [4] Cas J. F. Cremers. *Scyther - Semantics and Verification of Security Protocols*. Ph.D. dissertation, Eindhoven University of Technology, 2006.
- [5] Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–207, 1983.
- [6] Cédric Fournet and Martín Abadi. Hiding Names: Private Authentication in the Applied Pi Calculus. In *ISSS 2002*, volume 2609 of *LNCS*, pages 317–338. Springer, 2003.
- [7] Goichiro Hanaoka, Tsuyoshi Nishioka, Yuliang Zheng, and Hideki Imai. A hierarchical non-interactive key-sharing scheme with low memory size and high resistance against collusion attacks. *Comput. J.*, 45(3):293–303, 2002.
- [8] Gabriel Leen and Donal Heffernan. Expanding automotive electronic systems. *IEEE Computer*, 35(1):88–93, 2002.
- [9] Gavin Lowe. A hierarchy of authentication specification. In *CSFW '97*, pages 31–44. IEEE Computer Society, 1997.
- [10] Tsutomu Matsumoto and Hideki Imai. On the key predistribution system: A practical solution to the key distribution problem. In *CRYPTO 1987*, volume 293 of *LNCS*, pages 185–193. Springer, 1988.
- [11] C. Meadows. Open Issues in Formal Methods for Cryptographic Protocol Analysis. In *MMM-ACMS 2001*, volume 2052 of *LNCS*, pages 21, 2001.
- [12] Daisuke Nojiri, Goichiro Hanaoka, and Hideki Imai. A practical implementation of hierarchically structured key predistribution system and its evaluation. In *ISW 2000*, volume 1975 of *LNCS*, pages 224–236, 2000.
- [13] Hisashi Oguma, Akira Yoshioka, Makoto Nishikawa, Rie Shigetomi, Akira Otsuka, and Hideki Imai. New Attestation Based Security Architecture for In-vehicle Communication. In *IEEE Global Telecommunications Conference (GLOBECOM 2008)*, pages 1909–1914, 2008.
- [14] Pioneer USA - car navigation, Important: Notice of Free Firmware Update for AVIC F-Series, <http://www.pioneerelectronics.com/PUSA/Support/Navigation/Notice+of+Free+Firmware+Update+for+AVIC+F-Series>
- [15] Douglas R. Stinson. On some methods for unconditionally secure key distribution and broadcast encryption. *Des. Codes Cryptography*, 12(3):215–243, 1997.
- [16] Toyota recalls 160,000 Prius hybrids due to software glitch, <http://www.forbes.com/feeds/afx/2005/10/13/afx2276901.html>.
- [17] Trusted Computing Group. TPM Specification Version 1.2. <https://www.trustedcomputinggroup.org/downloads/specifications>.
- [18] Marko Wolf, André Weimerskirch, and Thomas Wollinger. State of the Art: Embedding Security in Vehicles. *EURASIP Journal on Embedded Systems*, 2007(Article ID 74706, 16 pages), 2007.