Physical Layer Group Key Agreement for Automotive Controller Area Networks

Shalabh Jain and Jorge Guajardo

Robert Bosch LLC, Research and Technology Center,
Pittsburgh, PA 15203, USA
{shalabh.jain, jorge.guajardomerchan}@us.bosch.com

Abstract. Distribution of cryptographic keys between devices communicating over a publicly accessible medium is an important component of secure design for networked systems. In this paper, we consider the problem of group key exchange between Electronic Control Units (ECUs) connected to the Controller Area Network (CAN) within an automobile. Typically, existing solutions map schemes defined for traditional network systems to the CAN. Our contribution is to utilize *physical properties* of the CAN bus to generate group keys. We demonstrate that *pairwise interaction* between ECUs over the CAN bus can be used to efficiently derive group keys in both authenticated and non-authenticated scenarios. We illustrate the efficiency and security properties of the proposed protocols. The scalability and security properties of our scheme are similar to multi-party extensions of Diffie-Hellman protocol, without the computational overhead of group operations.

Keywords: Automotive security; ECU keys; CAN bus; Group keys

1 Introduction

Modern automobiles, in conjunction with mechanical components, utilize several electronic components (ECUs) for sensing, actuation, user interface and control. The ECUs communicate over a shared medium known as the CAN bus. Over the past decade, several ECUs that connect to external networks, e.g. via a Bluetooth, cellular or a wired interface, have been added to the CAN bus. Such interfaces enable remote access to the components on the internal network of the car. While these may be utilized to enable several useful functions for the users, e.g. emergency messaging systems, remote ignition, they can easily be misused by a malicious attacker. The ease of attack, and damage due to adversarial behavior has been demonstrated by several researchers over the past few years in [6, 19, 22, 24, 28].

A common observation by the attackers in [6, 22], is the lack of security mechanisms in the CAN architecture. Traditional automobiles were designed as standalone systems, intended for autonomous operation. Thus, latency and reliability were the dominant criterion for network design. However, with increased connectivity, there is a need to secure the internal network from external attackers. The current automobile manufacturers utilize traditional network security

principles at the periphery (firewalls, access control), to secure the CAN access. However, as demonstrated recently in [28], these techniques may not offer sufficient protection. Further, such methods do not address the fundamental lack of security in CAN messages.

Generally speaking, the attacks demonstrated thus far may be roughly divided into two stages. First, the attackers compromise an ECU with a remote interface and the ability to inject arbitrary messages on the CAN bus. Secondly, the attackers communicate with a critical ECU over the CAN bus and influence its behavior. The second stage is enabled by the broadcast nature of the internal network and the lack of authentication. Typically, any operation in the second stage requires knowledge of the internal bus protocol and message structure, which has been simplified by the lack of encryption on the network.

It is clear that any security solution for CAN should include fundamental protections such as source authentication and packet level encryption. Several researchers, e.g. [8, 14, 26], have proposed methods to include these primitives in the current CAN architecture. One of the fundamental requirements to enable these primitives is the existence of cryptographic keys shared between the communicating ECUs. However, it is challenging to pre-install group keys during production of the ECU or securely manage the keys over the long lifetime of a vehicle. Thus, we require an efficient key generation and exchange protocol that can be executed during the operation of the car to agree on secret keys.

To ensure minimal disturbance to critical operations on the CAN bus, the key exchange protocol must be bandwidth efficient. Further, it must incur a low computational overhead to accommodate a variety of ECU capabilities. Since CAN messages are multicast, it is necessary for the protocol to support the generation and update of group keys. In this paper, we propose such a protocol by utilizing the physical properties of the CAN bus.

1.1 Our Contributions

We extend the two-party protocol proposed in [23] to the generalized group scenario. Our contributions are as follows,

- We utilize the physical properties of the CAN bus to construct a group key exchange protocol that is secure in the information-theoretic sense from eavesdroppers.
- For the restricted scenario of computationally bounded adversaries, we propose a highly efficient tree based structure for our protocol that has logarithmic complexity for node addition and deletion.
- We propose an efficient authenticated group key exchange protocols that utilizes only the pre-established trust between the individual ECUs and the gateway.

1.2 Related Work

CAN Security: In this work, we utilize the physical properties of the CAN bus for exchange of keys. To the best of our knowledge, the first to utilize such properties for key agreement are Müller and Lothspeich in [23]. Their work forms the

basis for our constructions and it will be reviewed in detail in Section 3. Security for CAN networks, particularly authentication and integrity of messaging, has been considered previously in [8,11,14,26,27]. However, this line of work assumes that a shared key already exists.

(Group) Key Agreement: Distribution of group keys for both authenticated and unauthenticated scenarios has been explored in literature for well over three decades. Several schemes have been proposed based on varying assumptions of adversarial behavior and initial setup. One of the earliest results in this direction, Diffie-Hellman (DH) key exchange [7], uses the hardness of computing discrete-log over prime order groups to generate keys between a pair of nodes. Steiner et al. in [25] proposed an extension of DH to groups that uses a mixture of point-to-point messages and broadcast messages. This was modified by authors in [16–18, 25], who utilize a tree based structure to improve communication efficiency and support efficient addition/deletion of nodes. Authors in [29] reduce communication and storage overhead by performing these group operations over elliptic curves.

Several methods have also been proposed to generate authenticated group keys, either by extension of the two-party protocols to groups or by using ideas based on secret-sharing, e.g. [2, 4, 12, 15]. These schemes have several desirable properties such as provable security, perfect forward secrecy (PFS) and key independence. Most schemes, e.g. [2,4,15,25,29], involve expensive group operations over prime fields, and thus are not suitable for computationally constrained devices on the CAN bus. Other protocols, e.g. [12], fail to provide security against an adversary that can compromise the pre-shared secrets. This property is desirable for automotive networks, where some nodes may be easily compromised due to open accessibility or lack of protections. Our protocol provides these security properties. Our main differentiation from these lies in utilization of the physical properties of the CAN bus as a substitute for the expensive operations.

1.3 Organization

The remainder of the paper is organized as follows. In Section 2, we describe the system assumptions and the adversarial model. We present the scheme from [23] in Section 3. We propose two extensions of this scheme that are secure against passive adversaries in Section 4. In Section 5, we propose two alternative protocols that provide cryptographic guarantees against active adversaries. We discuss the security and performance issues of our schemes in Section 6.

2 Preliminaries

2.1 Notation

We adhere to the following notation for the paper. We denote a random n bit value value x sampled uniformly from the set $\{0,1\}^n$, consisting of all possible binary strings of n bits, as $x \leftarrow \{0,1\}^n$. We denote by $x := y$, the assignment of the value y to x.

For a binary string $x \in \{0,1\}^*$, $|x|$ represents the length of string and x' represent the complement of the string. For an index set $L \subseteq \{1,\ldots,|x|\}$, $x(L)$

refers to the substring with indices in L. If L consists of a single element, $x(L)$ simply refers to the Lth bit. Given two strings $x, y \in \{0,1\}^*$, $x \mid\mid y$ denotes the concatenation of the strings.

We denote by $I(X \wedge Y)$, the mutual information between random variables X and Y. $I(X \wedge Y) = H(X) - H(X|Y)$, where $H(X)$ is the entropy of X.

2.2 System Model and Assumptions

We consider the typical automotive network, comprising of ECUs connected via a shared wired bus that acts as a broadcast medium. During arbitration, the CAN bus allows multiple nodes to write simultaneously to the bus and observe the *overlapped* bus output. This feature, typically used for contention resolution, is essential for our scheme. Note that the current ECU design allows simultaneous read and write only during the *arbitration phase*, and not in the data phase. We assume that the CAN controller is sufficiently modified to allow this functionality for the entire packet. This can be achieved either via hardware or software modifications.

The typical CAN bus has two logical states, the dominant '0' state, where the bus is driven by a voltage, and the recessive '1' state, where the bus is grounded. If two nodes transmit a bit simultaneously, the effective state of the bus is dominant '0' if any of the nodes transmits a dominant signal. Thus, the bus acts as a logical AND gate between inputs from the nodes. This property, identified by authors in [23] to share sequence of bits between a pair of nodes, forms the basis of our scheme and hence a central assumption for our work. Note that though this work is in context of automotive networks, the scheme can be applied to any wired bus architecture that exhibits this property.

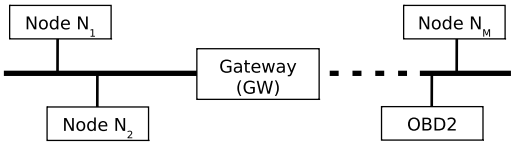

Fig. 1. Example of nodes connected by a shared medium

The typical CAN architecture, illustrated in Fig. 1, consists of one or more powerful nodes that act as gateway nodes (**GW**). As described by the authors in [22], current CAN architectures allow ECUs across different subnets to communicate transparently through the gateway. For our work, we assume each node shares a trust relationship with the gateway in the form of a pre-shared symmetric key. Thus nodeN_i shares key K_i with the gateway (**GW**). Such a relationship can be established during the vehicle manufacturing process, or during ECU installation by a mechanic.

For communication within a group, we assume a pre-existing communication ordering between the nodes. In typical automotive scenarios, different ECUs have well defined priority. Thus, the communication order may be pre-assigned based on ECU identity/priority, or assigned by random arbitration over the shared medium. Alternately, it can be defined in a common file, e.g. FIBEX file, and

shared with the ECUs. We assume that for any group configuration, the member ECUs can determine their communication order.

Our protocols are based on simultaneous transmission by two nodes. Thus, all interactions in our system are between ECU pairs. We refer to the node that is earlier in the communication order as the primary or initiator node and the other node as the secondary or responding node. CAN messages are organized in frames that consist of an identifier field followed by the data field. We assume that the initiator uses the identifier to specify information about the key exchange session (e.g. session identifier) and the responder uses this to initiate simultaneous transmission of its message in the data field transmission phase. This identifier allows the protocol to be resumed in case of interruption by a ECU transmitting a 'critical' message for the automobile.

2.3 Adversarial Model

Several adversarial models have been proposed for key exchange protocols in literature, e.g. CK model [5], or BR model [3]. It is typically assumed that the adversary can record all messages transmitted on the bus, modify them, or insert its own messages.

Here, we consider two adversarial scenarios. For the schemes in Section 4, we restrict the adversarial behavior to passive observations. This model, though unrealistic for typical networks, can be sufficient for all attacks on the CAN bus. This is due to the inherent robustness that the CAN bus provides to active adversaries. Detailed analysis for this is presented in Section 6.1.

For schemes in Section 5, we consider a powerful adversarial scenario, wherein the adversary has complete control over the protocol execution. There, we argue that our schemes provides cryptographic guarantees against such adversaries. Due to the dependence of our scheme on physical properties of the bus, an adversary with a high resolution oscilloscope may be able to obtain the keys by probing the bus. Further, since our scheme does not have a practical implementation yet, it has not been analyzed for timing or power side-channels. We consider such attacks outside the scope of this paper.

2.4 Cryptographic Assumptions

For our protocol, we assume the existence of an indexed family of pseudorandom functions (PRF) [9], defined as,

Definition 1. *For the security parameter n, the function $g : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is a family of pseudorandom functions, indexed by the first parameter that satisfies the following conditions,*

- *For a randomly selected index $k \leftarrow \{0,1\}^n$, the function efficiently maps an element from the domain $\{0,1\}^n$ to the range $\{0,1\}^n$.*
- ***(Security Condition)** For an adversary that runs in polynomial (in security parameter) time, the output of the PRF, where the first parameter is randomly selected, is indistinguishable from random.*

In practice, PRFs can be realized either via a block cipher or by a well-designed efficient hash function with a random seed (as the index) as a part of the input.

Further, we utilize the definition of computational entropy of a random variable X from [13], defined as $H_C(X) = k \iff \{X \sim_C Y \text{ and } H(Y) \geq k\}$, i.e. For a PPT process, X is computationally indistinguishable from a random variable Y with true entropy greater than k. Clearly the following Lemma follows from Definition 1.

Lemma 1. *For a randomly selected $k \leftarrow \{0,1\}^n$, let $X_i = g(k,i)$. Then $H_C(X_i) = n,\ \forall i \in \{0,1\}^n$. Further, for $I \subset \{0,1\}^n$, where $|I| = r$, $H_C(X_I) = rn$, where X_I denotes the concatenation of all $X_i,\ i \in I$.*

2.5 Security Definition

We define the security of a key exchange scheme Π using the information theoretic notion from [21] as follows,

Definition 2. *A key agreement protocol $\{S_A, S_B\} = \Pi^{A,B}(1^k)$ between two parties A and B results in secret key outputs S_A, S_B at the respective parties. If the protocol terminates correctly, we have $|S_A| = k$, otherwise $S_A = S_B = \emptyset$. The protocol can be said to be secure if the following hold*

P1: The keys derived at the end match, $Pr\,(S_A \neq S_B) = 0$.
P2: If the observations of the adversary is characterized as Z, we have $I(S_A \wedge Z) = 0$.
P3: The key has entropy $H(S_A) = k$.

Here, we make no assumptions about the computational capability of the adversary. We argue a similar notion can be defined for computationally bounded adversaries by replacing the quantities in Definition 2 with their computational equivalent.

Definition 3. *A key agreement protocol $\{S_A, S_B\} = \Pi^{A,B}(1^k)$ is ϵ-secure for computationally bounded passive adversaries if the Definition 2 holds using the notion of computational entropy, i.e. $I_C(S_A \wedge Z) \doteq H_C(S_A) - H_C(S_A \mid Z) \leq \epsilon$, and $H_C(S_A) \geq k - \epsilon$.*

3 Two party Plug-and-Secure (PnS) protocol

The wired AND property of the CAN bus was first utilized by the authors in [23] for key agreement between a pair of nodes. Since our protocols are based on their scheme, we briefly describe it here. Our notation here differs from the original work to maintain uniformity with the remainder of our protocols. For details about implementation issues and synchronization, the reader is encouraged to read [23].

Let PnS(1^n, nodeA, nodeB, f_A, f_B) denote the Plug-and-Secure protocol between nodeA and nodeB, where the security parameter 1^n denotes the length, n, of the shared secret key produced when the protocol terminates. We have chosen to parameterize the random number generation (RNG) by the nodes nodeA and nodeB, using the functions f_A and f_B respectively. This allows for

Function: $RetVal := \mathbf{f_x(1^n)},\ x \in \{A, B\}$

$r \leftarrow \{0,1\}^n$
$RetVal := r$

Protocol: **PnS(1^n, nodeA, nodeB, $\mathbf{f}_A$, $\mathbf{f}_B$)**

1. nodeA and nodeB initialize secret strings as null, i.e. $s_A, s_B := \emptyset$.
2. nodeA and nodeB obtain random values $a := f_A(1^n)$ and $b := f_B(1^n)$.
3. Both nodes simultaneously write a, b to the bus and observe the bus output y.
4. Next similarly they write a', b' to the bus and observe the bus output z.
5. Let $\mathcal{G} = \{1 \leq j \leq n \mid y(j) = 0\ AND\ z(j) = 0\}$. This represents the set of secret bits.
6. It can be easily verified that $a(\mathcal{G}) = b(\mathcal{G})'$. The primary node (nodeA here) sets $s_A := s_A \mid\mid a(\mathcal{G})$. The secondary node (nodeB here) sets the complementary bits, i.e. $s_B := s_B \mid\mid b(\mathcal{G})'$.
7. The string $s_A = s_B$ is shared secret between nodeA and nodeB and is the result of the subroutine.
   When used as a subroutine, the protocol halts here.
8. If the length of the string is insufficient, i.e. $|s_A| = |s_B| < n$, the protocol repeats from Step (2).
9. The final string $s_A = s_B$ is shared secret between nodeA, nodeB of length n.

Protocol 1. Two party PnS protocol from [23]

a uniform presentation of the protocols, while providing the flexibility to alter the instantiation of the RNG across them. This advantage will become more evident in the group protocols. The random number generators are private for each of the nodes and maintain an independent and persistent state through the protocol. This state is typically expressed as a counter that is initialized during the first execution and incremented upon successive executions. We illustrate the sequence of operations of the protocol from [23] as Protocol 1.

In the description of Protocol 1, we assumed that nodeA was the initiator (primary node) and nodeB was the secondary node. Each node discards bits that are leaked to the adversary. It can be seen from Step 5 that the bit positions where either y or z are 1, correspond to indices of strings a and b that can be determined by any eavesdropper. Thus they are no longer secret.

$$y(i) = 1 \Leftrightarrow a(i) = 1\ AND\ b(i) = 1,$$
$$z(i) = 1 \Leftrightarrow a(i) = 0\ AND\ b(i) = 0.$$

The messages to initiate the protocol, and the parameter negotiation to determine the desired key length are omitted here. Similarly, we do not specify the key verification approach at the end of the protocol. A number of existing initialization and verification techniques can be used with the protocols. Here, our goal is to present the fundamental building block that is the basis for our group key protocol.

It is clear from the description that Protocol 1 is very efficient and it does not require any expensive cryptographic operations. Further, it inherits the properties of contributory protocols such as Diffie-Hellman. This makes it highly suit-

able for the constrained ECU environments. Several properties of Protocol 1 will be inherited by our group key protocols. We illustrate a few key properties here.

Security: We demonstrate that Protocol 1 is secure against computationally unbounded *passive* adversaries.

Theorem 1. *The protocol* $\Pi^{A,B}_{Prot1}(1^n)$ *satisfies Definition 2, where* $\Pi^{A,B}_{Prot1}(1^n)$ *denotes Protocol 1 between nodeA and nodeB.*

Proof. Denote by s_A, the secret key at nodeA and by Y, Z, the complete observations of the adversary corresponding to Steps (3) and (4) respectively. The Property (P2) can be simply verified by the correctness of the protocol upon termination. Further, $H(s_A) = n$, as the samples were uniformly selected. Denote by L as the set of indices of the random values output on the channel that contributed to the bits of the key. Then we show Property (P2) as follows,

$$\begin{aligned} I(s_A \wedge \{Y, Z\}) &= I(s_A \wedge \{Y(L), Z(L)\}) && \textit{(bits are iid)} \\ &= nI(s_A(l_1) \wedge \{Y(l_1), Z(l_1)\}),\ l_1 \in L && \textit{(key bits are iid)} \\ &= n(H(s_A(l_1)) - H(s_A(l_1) \mid Y(l_1), Z(l_1))) = 0. \end{aligned}$$

Key Independence: Successive invocation of f_A and f_B produce independent random strings. Since each instance of key generation depends only on the outputs of f_A, f_B, the current key reveals no information about the past keys or future keys.

Function: $RetVal := \mathbf{f_x(1^n, s)},\ x \in \{A, B\}$

i: local persistent counter initialized to 0 during the first execution
$RetVal := g(s, i)$
$i := i + 1$

Protocol: **CompPnS($\mathbf{1^n}$, nodeA, nodeB, $\mathbf{f}_A$, $\mathbf{f}_B$)**
All steps of Protocol (1) remain the same except Step 2 which changes to the following
2(a). nodeA and nodeB compute local random values $t_a \leftarrow \{0,1\}^n, t_b \leftarrow \{0,1\}^n$.
2(b). The nodes obtain the random values for the protocol execution as
$a := f_A(1^n, t_a)$ and $b := f_B(1^n, t_b)$.
8 (new). The protocol repeats from 2(b) of the new protocol.

Protocol 2. Computational version of the two party PnS

Computational Definition: We define a computational version of Protocol 1 by using PRFs to generate the random values a and b. The changes required are briefly summarized in Protocol 2. This protocol can be proved to be secure against computationally bounded *passive* adversaries.

Theorem 2. *The protocol* $\Pi^{A,B}_{Prot2}(1^n)$*, satisfies Definition 3, where* $\Pi^{A,B}_{Prot2}(1^n)$ *denotes Protocol 2 between nodeA and nodeB.*

Proof. The proof follows from the proof of Theorem 1, by applying the computational definition of to conditional entropy. It will be included in the extended version of this paper.

4 Group Key Agreement Schemes

In this section we introduce two new group key agreement protocols without authentication. Though these can be viewed as a special case of the authenticated protocol, they warrant separate treatment due to different complexity and security properties.

The protocols presented here require a linear (in size of the group) number of interactions for initial key establishment. Intuitively, the broadcast nature of the CAN bus allows pairwise PnS interaction between successive nodes to be sufficient for global key agreement. Once two nodes execute the PnS protocol, they may be viewed as a single logical entity for any further transmissions by one of these nodes, based the PnS output. Thus, each successive interaction increases the size of the logical entity by one, until the whole group is created.

For the remainder of this paper, we assume that the group consists of M nodes, $\{\text{nodeN}_1, \ldots, \text{nodeN}_M\}$. For simplicity, we assume the communication sequence to be based on the lexicographic order, i.e. nodeN_1-nodeN_2-...-nodeN_M. We assume that the protocol initiation is triggered by the gateway node with information about the group members and parameters. The ECUs can determine their communication priority in a distributed manner based on the group configuration.

4.1 Simple Group Protocol

We first consider the simple extension of Protocol 1 to the M node scenario. The flow of messages to agree on the group key is illustrated in Protocol 3. The correctness of the protocol can be understood by examining Step (3) of Protocol 3. The first time this step is executed between nodeN_2 and nodeN_3, there exists a shared secret t_{N_2} between nodeN_1 and nodeN_2. Since t_{N_2} is the only value used by nodeN_2 in the PnS execution, the view of nodeN_1 is the same as nodeN_2. Thus the secret shared by PnS execution between nodeN_2 and nodeN_3 can be computed by nodeN_1 independently, based on the observed bus outputs. Also note that the bits of t_{N_3} obtained at the end of this step is a subset of the bits of t_{N_1}, i.e. $\exists I \subseteq \{1, \ldots, |t_{N_1}|\},\ s.t\ t_{N_3} = t_{N_1}(I)$.

For each repetition of this step between successive pairs of nodes, all nodes prior to the active pair can derive the result of the protocol. Thus, once nodeN_M completes execution, all nodes share a common string. Though the implicit backward-sharing of keys is a desirable property, the overall communication efficiency of the protocol is low. To see this, observe that at each each successive execution of Step 3, additional bits are leaked to an eavesdropper. Consider the ith bit of the string sampled by nodeN_1, $a_{N_1} := f_{N_1}(1^n, \emptyset)$. The probability that this bit does not leak by the end of the protocol is simply $2^{-(M-1)}$. Thus we obtain that the expected communication complexity to generate a n bit secret is exponential in the number of group elements, i.e. $\mathcal{O}(n \cdot 2^{(M-1)})$.

Node Arrival and Departure: At the end of the protocol, each node knows all random bits selected during the protocol. Thus, the departure of any node

Function: $RetVal := \mathbf{f_{N_i}(1^n, s)},\ 1 \leq i \leq M$

if $s \neq \emptyset$, $RetVal := s$
otherwise $r \leftarrow \{0,1\}^n$, $RetVal := r$

Protocol: **SimpleGroup($\mathbf{1}^n$, {nodeN$_1$, ..., nodeN$_M$}, $\{f_{N_1}, \ldots, f_{N_M}\}$)**

1. Each node initializes the secret key string $s_{N_i} = \emptyset,\ 1 \leq i \leq M$.
2. The first pair of nodes executes PnS *subroutine* with a target string length $l = n \cdot 2^{(M-1)}$, to obtain the shared secret as

$$t_{N_{1,2}} := \text{PnS}(1^l, \text{nodeN}_1, \text{nodeN}_2, f_{N_1}(1^l, \emptyset), f_{N_2}(1^l, \emptyset)).$$

   Each node maintains the temporary value of the PnS as $t_{N_1} = t_{N_2} := t_{N_{1,2}}$ respectively.
3. The next pair of nodes (nodeN$_2$, nodeN$_3$) executes PnS with the target length $l = |t_{N_2}|$ to obtain the private results (or update the private results) as

$$t_{N_{2,3}} := \text{PnS}(1^l, \text{nodeN}_2, \text{nodeN}_3, f_{N_2}(1^l, t_{N_2}), f_{N_3}(1^l, \emptyset)).$$

4. All nodes prior to the currently active nodes update their private strings as the output of the PnS result. In this case, as nodeN$_1$ is the only node preceding (nodeN$_2$, nodeN$_3$), it updates its private string t_{N_1} as the output of the PnS protocol between nodeN$_2$ and nodeN$_3$. Thus $t_{N_1} = t_{N_2} = t_{N_3}$.
5. Protocol is repeated from Step (3) for each successive pair of nodes (nodeN$_3$, nodeN$_4$), ..., (nodeN$_{M-2}$, nodeN$_{M-1}$), (nodeN$_{M-1}$, nodeN$_M$).
6. All nodes update the shared keys as $s_{N_i} = s_{N_i} \mid\mid t_{N_i},\ 1 \leq i \leq M$.
7. If $|s_{N_M}| < n$, the protocol is repeated from Step (2) using $l = (n - |s_{N_M}|) \cdot 2^{(M-1)}$.

Protocol 3. Simple group key protocol for M nodes

requires re-execution of the complete protocol. For node arrival, it may appear that the new node can simply be appended to the end of the chain. However, the execution of PnS with the new node would leak several bits (half on average). Thus, the whole protocol needs to be re-executed to compensate for the lost bits. Thus, both addition and deletion operations incur exponential communication cost, i.e. $\mathcal{O}(n \cdot 2^{(M-1)})$. However, the new key maintains the property of key independence.

Note that an alternative, more efficient protocol using PnS with information theoretic security guarantees could be envisaged if we do not require the protocol to be contributory, i.e. each node contributes to the randomness of the key. A selected leader can simply engage in pairwise PnS with all other nodes and use the derived keys as one-time pads to distribute a secret value. It is easy to see that the computational complexity for key generation, node departure and arrival for such a scheme would be linear, i.e. $\mathcal{O}(n \cdot M)$. However all protocols presented here are 'contributory' protocols.

Security: Since Protocol 3 simply extends Protocol 1, it has the advantage of inheriting the security properties of Protocol 1. As each stage of the protocol is

secure against *passive* adversaries, the information theoretic security extends to the whole protocol. Similarly, it can be simply observed that the key independence property extends from each stage to the eventual protocol result.

4.2 Tree Based Group Protocol

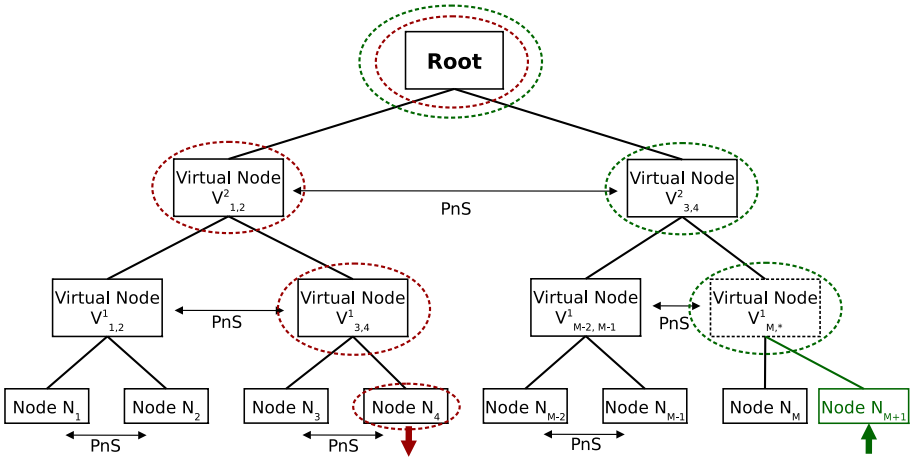

Fig. 2. Tree structure of PnS operations

The scheme presented in Section 4.1 provides ideal security guarantees at the cost of efficiency. However, security against computationally bounded adversaries is sufficient for practical systems. This relaxation enables the utilization of efficient topologies for key agreement.

For key generation, the nodes are organized in a binary tree structure, e.g. as shown in Fig 2. The physical nodes (ECUs) are assigned to the leaf nodes of the tree. The virtual nodes correspond to logical entities that can be emulated by *any* physical leaf node in the subtree rooted at that node. For the algorithms in this paper, we assume that the physical messages triggered by the virtual node are sent by the leaf node in the subtree with the highest priority (leftmost node of the tree in our model). The message flow for the key generation scheme is detailed in Protocol 4.

The structure of the scheme is similar to the previous protocol. However, using the function $g(\cdot,\cdot)$ isolates successive PnS stages. Since the output of $g(\cdot,\cdot)$ is indistinguishable from random, it can be used in place of the random sampling in the original protocol. Secondly, as the leakage of output bits of $g(\cdot,\cdot)$ leaks no information about the inputs, bits leaked at any stage do not influence the prior stages. As a result of these properties, this scheme incurs a linear communication overhead for initial key generation, i.e. $\mathcal{O}(n \cdot M)$. The tree structure further optimizes node addition and deletion.

Node Departure: A node in the network has knowledge of all the random values generated and exchanged along the path, denoted as $\mathcal{P}_{dr}$, from the node

Function: $Retval := \mathbf{f_{N_i}(1^n, s)},\ 1 \leq i \leq M$

i: local persistent counter initialized to 0 during the first execution
$RetVal := g(s, i)$
$i := i + 1$

Protocol: **TreeGroup(1^n, {nodeN$_1$, ..., nodeN$_M$}, $\{f_{N_1}, \ldots, f_{N_M}\}$)**

1. Each leaf node initializes the private string $t_{N_i} \leftarrow \{0,1\}^n,\ 1 \leq i \leq M$.
2. The process starts at the leaf nodes. Each pair of siblings execute the complete PnS protocol with target string length n, and the result is assigned to the private string of the parent as
$$t_{V^1_{i,i+1}} := \text{PnS}(1^n, \text{nodeN}_i, \text{nodeN}_{i+1}, f_{N_i}(1^n, t_{N_i}), f_{N_{i+1}}(1^n, t_{N_{i+1}})),$$
where $i = 1, 3, \ldots$. Note: Here, we execute the complete PnS protocol. Thus the output is of length n, i.e. $|t_{V^1_{i,i+1}}| = n$.
3. Step (2) is repeated at the next level of hierarchy, i.e. first level of virtual nodes here, $V^1_{i,i+1}$, to generate the private strings for their parents.
4. The process of Step (3) continues till the virtual root node is reached. The private string of the root node t_{root} is the shared secret key between all nodes.

Protocol 4. Tree based group key protocol for M nodes

to the root. Thus deletion of a node involves updating all the values known to the node and re-execution of PnS with the updated values. For example in Fig 2, if nodeN$_4$ departs the network, it is sufficient to update the random values at nodeN$_3$ and the virtual nodes $V^1_{3,4}, V^2_{1,2}$, root.

We assume that the departing node broadcasts its identity to the group. Thus the nodes along $\mathcal{P}_{dr}$ and their siblings flag their values for updating. The update progresses upwards from the affected leaf node. If a node lies directly along $\mathcal{P}_{dr}$, it uses the new PnS result from the child node for all future protocol execution. All other nodes simply execute the PnS protocol with updated index values (in $f(\cdot)$).

At the end of the protocol, the value of the final PnS interaction is used as the group secret shared by all nodes. The statistical independence of the output of $g(\cdot, \cdot)$ for different inputs ensures that the new key is independent of the prior shared sequence and unknown to the departing node. Further, it can be observed that the computational complexity of this stage is simply $\mathcal{O}(n \cdot \log M)$.

Node Arrival: Similar to the node departure scenario, a node arrival requires creation of a path to the root and executing the PnS protocol with siblings of the nodes along the new path.

For simplicity, we assume that the new node is temporarily assigned the priority equivalent to a recently departed node or the lowest priority among existing nodes in the group. This minimizes the changes to the tree structure and the re-computations required to add a node. In cases where this is not possible, we may add a node in the 'pre-assigned' order and modify the tree hierarchy accordingly.

Consider the example in Fig 2, where nodeN_{M+1} joins the network. This requires updating the random values at nodeN_M and the virtual nodes $V^1_{M,M+1}, V^2_{3,4}$, root. This may be performed in a manner identical to the departing scenario. Thus, it can be observed that the new key will be independent of the old key and the computational complexity is simply $\mathcal{O}(n \cdot \log M)$.

Security Discussion: The organization of the nodes in a tree structure does not alter the role of the adversary. It simply modifies the order of participation of the nodes. Intuitively, we argue that similar to Protocol 3, the security of the series of PnS stages against a computationally bounded adversary can be derived directly from Theorem 2.

5 Authenticated Group Key Agreement Schemes

We now consider the scenario where group members must be authenticated prior to participation in the protocol. This requires some pre-established notion of trust or identity that can be verified. As described in Section 2.2, we consider the minimalistic scenario where each node shares a symmetric key with the gateway.

In our schemes, the gateway simply acts as a passive verifier of the operations. The broadcast channel allows the gateway to monitor the protocol execution. We ensure that the messages are a function of the shared keys K_i's. This allows the gateway to verify whether the messages used for the PnS protocol are from the expected parties. We present two implementations of this approach that provide a tradeoff between security and efficiency.

5.1 Authenticated Tree Based Protocol

First we utilize the efficient tree structure of Protocol 4 and add an authentication mechanism to it. This can be achieved via a simple modification to the method of selection of random values by the leaf nodes. In Protocol 4, the leaf nodes choose arbitrary random values for the initial sequence of PnS operations. Here, we assume that **GW** provides a random value and all nodes use a function of this random value and their shared key to bootstrap the PnS procedure.

Since the **GW** is aware of the random value and the shared keys, it can recreate and thus verify all random strings used in the PnS operations. Note that after the initiation of the protocol by **GW**, it only participates passively. If an error is detected, the **GW** halts the protocol by transmitting an error message. The detailed flow of messages for this is presented in Protocol 5.

As the structure of the protocol is similar to Protocol 4, it inherits the low complexity and security properties of the unauthenticated protocol. However, authenticated key exchange protocols may have an additional security requirement of Perfect Forward Secrecy (PFS), wherein the group key remains secret even in the event of compromise of trust credentials, i.e. K_i. Protocol 5 however fails to meet this requirement. In the event that an adversary compromises the shared secret, it can reconstruct the random values used for PnS (similar to **GW**) and hence learn the secret key from the transcripts. In the next section, we provide a solution to this problem.

Function: $RetVal := \mathbf{f_{N_i}}(\mathbf{1^n}, \mathbf{s}),\ 1 \leq i \leq M$

i: local persistent counter initialized to 0 during the first execution
Output $RetVal := g(s, i)$
$i := i + 1$

Protocol: **AuthTreeGroup(1^n, {nodeN$_1$, ..., nodeN$_M$}, $\{f_{N_1}, \ldots, f_{N_M}\}$)**

1. **GW** select a random sequence of n bits, i.e. $t_{gw} \leftarrow \{0,1\}^n$ and broadcasts it to all group members.
2. Each leaf node of the tree initializes the private string $t_{N_i} = g(K_i, t_{gw}),\ 1 \leq i \leq M$. Here, K_i is the key shared between nodeN$_i$ and the **GW**.
3. The process starts at the leaf nodes. Each pair of siblings execute the PnS protocol with target string length n, and the result is assigned to the private string of the parent as
$$t_{V^1_{i,i+1}} := \text{PnS}(1^n, \text{nodeN}_i, \text{nodeN}_{i+1}, f_{N_i}(1^n, t_{N_i}), f_{N_{i+1}}(1^n, t_{N_{i+1}})),$$
where $i = 1, 3, \ldots$. Note that we execute the complete PnS protocol here so that the output is of length n, i.e. $|t_{V^1_{i,i+1}}| = n$.
4. Step (3) is repeated at the next level of hierarchy, i.e. first level of virtual nodes here, $V^1_{i,i+1}$, to generate the private strings for the parents of the virtual nodes.
5. The process of Step (4) continues till the virtual root node is reached. The private string of the root node t_{root} is the shared secret key between all nodes.
6. The gateway monitors the broadcast messages and verifies the correctness. It transmits an error message if the verification fails at any stage.

Protocol 5. Authenticated tree based group key protocol for M nodes

5.2 Authenticated Linear Group Protocol

In Protocol 5, all inputs used for computing the random values for PnS were available to an adversary that compromises K_i. This was because one of the inputs was broadcast by **GW** to initiate the protocol. Here instead of the broadcast message, the **GW** transfers the initial random value to nodeN$_1$ through the PnS protocol. This ensures that *atleast* one of the inputs is never leaked to the adversary. However, as this requires the secret to be passed down the authentication chain, it forces us to use a linear structure. The flow of the protocol is illustrated as Protocol 6.

Whenever two nodes engage in the PnS protocol, the first node uses a function of the random value from the previous stage, while the second node uses a fresh random value concatenated with some authentication credentials. The value used by the first node ensures that all nodes prior to it can re-create the PnS execution and learn its outputs. The value of the second node will be authenticated by the passively monitoring **GW**, before it is included in the chain.

To ensure security against compromise of K_i and still ensure verifiability, it is required that the second node use some fresh randomness, unknown to everyone else and the key K_i. It should be observed that successful authentication of the messages of the second node requires the PnS protocol to be internally executed atleast twice. In the first round, the fresh random value is extracted by **GW**

Function: $RetVal := \mathbf{f_x(1^n, K, ctr, state})$, $x \in \{\mathbf{GW}, \text{nodeN}_1, \ldots, \text{nodeN}_M\}$

i: local persistent counter initialized to 0 during the first execution
l_state: local persistent flag initialized to 0 during the first execution
if($l\_state \neq state$)
  if($state == 2$) $i := -1$
  if($state == 1$) $i := 0$
  $l\_state := state$
if($i == -1$) $RetVal := ctr$
else $RetVal := g(K, ctr + i)$
$i := i + 1$

Protocol: **AuthLinearGroup(1^n, {nodeN$_1$, ..., nodeN$_M$}**, $\{f_{N_1}, \ldots, f_{N_M}\}$**)**

1. The **GW** is begins the protocol by acting as the first link in the PnS chain. The **GW** chooses $t_{GW} \leftarrow \{0,1\}^n$ and nodeN$_1$ chooses $t_{N_1} \leftarrow \{0,1\}^n$ to execute the PnS protocol as
$$t_{GW,N_1} = \text{PnS}(1^n, \mathbf{GW}, \text{nodeN}_1, f_{GW}(1^n, t_{GW}, 0, 1), f_{N_1}(1^n, K_{N_1}, t_{N_1}, 2)).$$
2. Next, nodeN$_2$ chooses a random value $t_{N_2} \leftarrow \{0,1\}^n$. nodeN$_1$ performs PnS with nodeN$_2$ as
$$t_{N_1,N_2} = \text{PnS}(1^n, \text{nodeN}_1, \text{nodeN}_2, f_{N_1}(1^n, t_{GW,N_1}, 0, 1), f_{N_2}(1^n, K_{N_2}, t_{N_2}, 2)).$$
3. Step (2) is repeated between successive pairs of nodes till the final node is reached. Denote by t_{N_{M-1},N_M}, the result of the final PnS operation. This is the group key shared by all nodes.
4. The gateway monitors the broadcast messages and verifies the correctness. It transmits an error message if the verification fails at any stage.

Protocol 6. Authenticated linear group key protocol for M nodes

and in the second round it is authenticated. We argue that this will always be the case as the probability that the PnS protocol is executed only once is 2^{-n}, i.e. when all bits of both the parties are complements of each other. Thus the authentication process does not add communication overhead. Similar to the previous schemes, the initial key generation has linear complexity, i.e. $\mathcal{O}(n \cdot M)$.

Node Arrival and Departure: The addition of a node in a linear structure is simple. We assume that the added node is temporarily (for the group) assigned a lower priority than all other elements and thus, is to be added at the end of the chain. Thus the addition of a new member simply requires one PnS operation between the last node and the new member, i.e. complexity of $\mathcal{O}(1)$.

A departing node knows the secrets associated with all nodes that follow it. Thus a node departure requires all nodes following the departing node to update their key parameters. This may be performed by simply re-executing the PnS protocol with updated index values, without the need of sampling fresh random strings. Thus this incurs linear communication cost, i.e. $\mathcal{O}(n \cdot M)$.

6 Discussion

6.1 Security Properties

Though Section 5 describes schemes that are robust against arbitrary active adversaries, we argue that such an adversarial model is too restrictive for the automotive scenarios. Operations of our protocol and the architecture of the CAN bus restrict the actions of the adversary in our system. We argue that an active adversary cannot successfully perform any operation, except eavesdropping, without detection. Consider the following

1. **Modification of a packet -** The properties of the CAN bus allow only one type of modification to the messages transmitted by the nodes. An adversary can flip a recessive bit '1' to a dominant bit '0' by transmitting a voltage, however not vice-versa. It can be verified that this simply results in a mismatched key at both parties. This can easily be detected by any key verification method.
2. **Inserting messages for active nodes -** An active node, executing a pairwise session of the protocol, only accepts outputs on the bus that result from superposition of its own signals with that of the partner. Thus consider an adversary that attempts to compromise a session between nodeN_1 and nodeN_2 by inserting a 'specific' message for nodeN_2. However, this requires that the adversary initiate a transmission from nodeN_2. Assume that the message transmitted by the adversary is m_{adv}, and that by nodeN_2 is m_{N_2}. Thus the message recorded by nodeN_2 is the logical AND of these messages, i.e. $m_{adv} \wedge m_{N_2}$. However, as the adversary has no control over m_{N_2}, it cannot insert a 'specific' packet. It can however choose and force bits to be 0. This can be detected by key verification.
3. **Inserting messages for passive nodes -** In the group protocols, nodes that have engaged in one pairwise session may update their local parameters based on the output of the future sessions. An adversary may falsely emulate such sessions. However, it can be demonstrated that the probability of 'successfully' inserting a n bit packet, i.e. a packet that is accepted as a valid input by the passive node, is less than $\left(\frac{3}{4}\right)^n$.

   Theorem 3. *Let the adversary activate the protocol of a passive node by inserting an arbitrary pair of strings b_1, b_2, where $|b_1| = |b_2| = n$, marked with the session identifier of the currently active nodes. The passive nodes detect the adversary with a probability greater than $1 - \left(\frac{3}{4}\right)^n$.*

   Proof. Consider the scenario where nodeN_2 and nodeN_3 are actively engaging in PnS and nodeN_1 is the passive observer. Let $t_{N_{1,2}}$ be the string at nodeN_1 as a result of its interaction with nodeN_2. As described in Protocol 1, nodeN_2 uses that string for interaction with nodeN_3. Thus nodeN_1 can simply verify the bus output to and identify 'unexpected' behavior of the adversary as follows. Consider the set of indices L where $t_{N_{1,2}} = 0$, $L = \{l \leq |t_{N_{1,2}}| | t_{N_{1,2}}(i) = 0\}$.
   The output on the bus as a result of the first PnS operation, corresponding

to indices in the set L, should be 0. This results simply from the *AND* operation of the bus. Any deviation from this results in an error by nodeN_1. Thus for the message by adversary to be accepted, $b_1(L)$ should be 0, i.e. the adversary should be able to estimate the position of all 0s in the string $t_{N_{1,2}}$. Thus we obtain

$$Pr\left(\{b_1, b_2\} \text{ accepted}\right) = \sum_k Pr\left(\text{Adv covers all 0 positions} \mid |L| = k\right) \cdot Pr(|L| = k)$$
$$= \sum_{k=0}^{n} \left(\frac{1}{2}\right)^k \cdot \binom{n}{k} \left(\frac{1}{2}\right)^k \left(\frac{1}{2}\right)^{n-k} = \left(\frac{3}{4}\right)^n$$

4. **Impersonation -** The broadcast nature of the CAN bus ensures that any transmitted message is delivered to all the nodes. Thus any spoofed or replayed message by the adversary can be detected by the victim node and an error flag can be raised. We assume that such detection can occur due to the session IDs described earlier.

It is clear that Properties 1, 2, 3 are guaranteed by any PnS based key agreement scheme for the CAN bus. A cryptographic method to guarantee Property (4) is by utilizing the trust relation established with the gateway. An alternate way is to increase ECU robustness and include a mechanism to identify spoofed messages in the individual ECUs. For such cases, schemes that are secure against a passive eavesdropper would also be secure against an active adversary. Thus the efficient tree-based structure of Section 4 can be utilized to provide security against active adversaries.

6.2 Performance

One of the main benefits of the our approach is its computational advantage over the modular multiplications, as required for group schemes based on DH or ECDH. The variants of the our protocols allow a tradeoff between the complexity and bandwidth. Further, our schemes are based on pseudorandom functions, which can be practically implemented either via the SHA family of hash functions or a block cipher such as AES. Both these primitives are better suited for resource constrained devices, compared to modular multiplication.

To understand the performance comparison of our scheme, consider the scenario where the M nodes wish to generate a n bit key. Clearly, Protocol 3 requires no cryptographic primitives, but has a high bandwidth overhead. Each round of PnS using n bit inputs requires transmission of $2n$ bits on the bus (normal and the complement). Further, scenarios that use the cryptographic primitives use 2 invocations of the function for each round. We summarize the overhead and some properties of the protocols in Table 1. Authors in [10] evaluate the performance of various cryptographic primitives on various automotive microcontrollers, namely the S12X, a low end 16-bit automotive microcontroller from Freescale and the TriCore chip, a high end 32-bit microcontroller from the

Table 1. Performance of the proposed schemes

Property	Protocol			
	3 Simple Unauth	4 Tree-based Unauth	5 Tree based Auth	6 Linear Auth
Avg no. of bits Tx on bus	$4n(2^{M-1}-1)$	$4n(M-1)$	$4n(M-1)$	$4nM$
Avg. of PRF invocations	0	$4(M-1)$	$5M-4$	$4M-2$
Node addition	$\mathcal{O}(n2^M)$	$\mathcal{O}(n\log M)$	$\mathcal{O}(n\log M)$	$\mathcal{O}(n)$
Node deletion	$\mathcal{O}(n2^M)$	$\mathcal{O}(n\log M)$	$\mathcal{O}(n\log M)$	$\mathcal{O}(nM)$

AUDO family of Infineon. The S12X family operates at 40MHz while the Tricore chips can operate up to 180MHz. For generating a key of length $n = 128$, we may utilize the SHA-256 hash function in place of the PRF. It can be seen that performance of the PRFs adds very little overhead of $3.145ms$ and $0.045ms$ respectively for each invocation for our target input lengths.

Due to the lack of implementation of state-of-the art key exchange schemes based on ECDH, e.g. MQV [20], on comparable automotive microcontrollers, we cannot present performance benchmarks for the group operations. However, to the best of our knowledge, in all performance benchmarks in literature, the group operations for ECDH with a comparable keysize (256 bits), implemented in software without any dedicated hardware support, consume atleast one to two orders higher time. Thus, we would assume that such a scaling would be expected for the automotive microcontrollers as well, i.e. overhead of $\approx 100ms$. Thus the performance advantage of our scheme is clear.

A typical implementation of the CAN bus operates at a rate of 125kbits/s, i.e. 1000 frames per second. Each frame of frame contains 8 bytes of data. Thus on average, a PnS session between two nodes to generate a $n = 128$ bit key would last over 8 frames, i.e. $8ms$, which is similar to the overhead using ECDH.

With improvements in technology, it is expected that the recommended length of parameters for ECDH will increase significantly in the foreseeable future, [1]. The computational overhead of group operations due to such changes has poor scaling in comparison to our group key protocols, which would scale linearly with the key length. Further, as physicists bring quantum computing into the practical realm, traditional EC-DH based schemes may be rendered insecure. By contrast, PnS based schemes would remain secure.

6.3 Conclusion

We presented methods to efficiently generate group keys for nodes connected via a shared bus, using physical layer properties. The methods utilize the natural wired AND operation provided by the bus architecture, in place of expensive modular exponentiation operations typically required. The algorithmic complexity of our scheme is equivalent to the most efficient key-agreement algorithm available today. One of the most promising applications for this schemes is in context of ECU networks inside an automobile. However, our assumptions are sufficiently generic to map to a variety of wired networks. Thus, these schemes can be utilized in different systems where low capability devices are present.

References

1. Cryptographic key length recommendations. http://www.keylength.com, accessed: 2016-02-09
2. Ateniese, G., Steiner, M., Tsudik, G.: Authenticated group key agreement and friends. In: Proc. of Conf. on Computer and Communications Security. pp. 17–26. ACM, New York, NY, USA (1998)
3. Bellare, M., Rogaway, P.: Entity authentication and key distribution. In: Stinson, D. (ed.) Advances in Cryptology - Crypto93, Lecture Notes in Computer Science, vol. 773, pp. 232–249. Springer Berlin Heidelberg (1994)
4. Bresson, E., Chevassut, O., Pointcheval, D.: Provably secure authenticated group diffie-hellman key exchange. ACM Trans. Inf. Syst. Secur. 10(3) (Jul 2007)
5. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) Advances in Cryptology - EUROCRYPT 2001, Lecture Notes in Computer Science, vol. 2045, pp. 453–474. Springer Berlin Heidelberg (2001)
6. Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S., Koscher, K., Czeskis, A., Roesner, F., Kohno, T.: Comprehensive experimental analyses of automotive attack surfaces. In: Proc. of USENIX Security Symposium (Aug 2011)
7. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (Nov 1976)
8. Glas, B., Guajardo, J., Hacioglu, H., Ihle, M., Wehefritz, K., Yavuz, A.: Signal-based automotive communication security and its interplay with safety requirements. Embedded Security in Cars (ESCAR) Europe (Nov 2012)
9. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM 33(4), 792–807 (Aug 1986)
10. Groza, B., Murvay, S.: Efficient protocols for secure broadcast in controller area networks. IEEE Transactions on Industrial Informatics 9(4), 2034–2042 (Nov 2013)
11. Groza, B., Murvay, S., Herrewege, A., Verbauwhede, I.: Proc of. Intl. Conf. on Cryptology and Network Security, chap. LiBrA-CAN: A Lightweight Broadcast Authentication Protocol for Controller Area Networks, pp. 185–200. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
12. Harn, L., Lin, C.: Authenticated group key transfer protocol based on secret sharing. IEEE Transactions on Computers 59(6), 842–846 (June 2010)
13. Hastad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM J. Comput. 28(4), 1364–1396 (Mar 1999)
14. Herrewege, A.V., Verbauwhede, I.: CANAuth - A Simple, Backward Compatible Broadcast Authentication Protocol for CAN bus. In: ECRYPT Workshop on Lightweight Cryptography 2011. pp. 229–235. Louvain-La-Neuve,BE (2011)
15. Katz, J., Yung, M.: Scalable protocols for authenticated group key exchange. In: Boneh, D. (ed.) Advances in Cryptology - CRYPTO 2003, Lecture Notes in Computer Science, vol. 2729, pp. 110–125. Springer Berlin Heidelberg (2003)
16. Kim, Y., Perrig, A., Tsudik, G.: Group key agreement efficient in communication. IEEE Transactions on Computers 53(7), 905–921 (July 2004)
17. Kim, Y., Perrig, A., Tsudik, G.: Communication-efficient group key agreement. In: Proc. of Annual Working Conf. on Information Security. pp. 229–244 (2001)
18. Kim, Y., Perrig, A., Tsudik, G.: Tree-based group key agreement. ACM Trans. Inf. Syst. Secur. 7(1), 60–96 (Feb 2004)

19. Koscher, K., Czeskis, A., Roesner, F., Patel, S., Kohno, T., Checkoway, S., McCoy, D., Kantor, B., Anderson, D., Shacham, H., Savage, S.: Experimental security analysis of a modern automobile. In: Proc Symposium on Security and Privacy. pp. 447–462 (May 2010)
20. Law, L., Menezes, A., Qu, M., Solinas, J., Vanstone, S.: An efficient protocol for authenticated key agreement. Des. Codes Cryptography 28(2), 119–134 (Mar 2003)
21. Maurer, U.: Information-theoretically secure secret-key agreement by not authenticated public discussion. In: Fumy, W. (ed.) Advances in Cryptology - EUROCRYPT. Lecture Notes in Computer Science, vol. 1233, pp. 209–225. Springer-Verlag (May 1997)
22. Miller, C., Valasek, C.: A survey of remote automotive attack surfaces. Tech. rep., IOActive Inc., [Online Whitepaper: Accessed 2016-02-09]
23. Müller, A., Lothspeich, T.: Plug-and-secure communication for CAN. CAN Newsletter pp. 10–14 (Dec 2015)
24. Rouf, I., Miller, R.D., Mustafa, H.A., Taylor, T., Oh, S., Xu, W., Gruteser, M., Trappe, W., Seskar, I.: Security and privacy vulnerabilities of in-car wireless networks: A tire pressure monitoring system case study. In: Proc. USENIX Security Symposium. pp. 323–338 (Aug 2010)
25. Steiner, M., Tsudik, G., Waidner, M.: Key agreement in dynamic peer groups. IEEE Transactions on Parallel and Distributed Systems 11(8), 769–780 (2000)
26. Szilagyi, C., Koopman, P.: Low cost multicast authentication via validity voting in time-triggered embedded control networks. In: Proc. Workshop on Embedded Systems Security. ACM, New York, NY, USA (2010)
27. Szilagyi, C., Koopman, P.: Flexible multicast authentication for time-triggered embedded control network applications. In: Proc. Intl. Conf. on Dependable Systems and Networks. pp. 165–174. IEEE (Jun 2009)
28. Valasek, C., Miller, C.: Remote exploitation of an unaltered passenger vehicle. Tech. rep., IOActive Inc., [Online Whitepaper: Accessed 2016-02-09]
29. Wang, Y., Ramamurthy, B., Zou, X.: The performance of elliptic curve based group diffie-hellman protocols for secure group communication over ad hoc networks. In: Proc Intl. Conf. on Communications. vol. 5, pp. 2243–2248 (June 2006)