A Tale of Two Automotive Security Services: A Formal Analysis

Teri Lenard

The Doctoral School of Letters, Humanities and Applied Sciences, "George Emil Palade" University of Medicine, Pharmacy, Science, and Technology of Targu Mures, 540139 Targu Mures, Romania,
Centre Universitaire d'Informatique, Geneva School of Economics and Management, University of Geneva, Route de Drize 7, CH-1227 Carouge, Switzerland
teri.lenard@unige.ch

Abstract. Automotive system faced in the past decade an abundance of security services proposed by the scientific literature to strengthen their system security. The solutions solve problems in terms of key distribution, data authentication, or system monitoring. While the volume of research done brings in consequence novel ideas, strong validation and extensive experimentation is a must to prove their viability and correctness. Consequently, the work at hand offers a formal analysis of two existing security services for automotive systems, namely for a Key Distribution Service (KDS) and for a data authentication and aggregation method titled Mixed data authentication for Controller Area Network (MixCAN). While the KDS aims to distribute long-term and short-term cryptographic keys, MixCAN envisions a lightweight authentication protocol through Encrypted Bloom Filters (EBFs). The objective of the formal analysis is to prove the correctness of the mentioned security solutions through a Burrows–Abadi–Needham (BAN) logic analysis.

Keywords: data authentication, key distribution, automotive systems, Burrows–Abadi–Needham (BAN) logic, formal analysis

1 Introduction

The field of automotive system security received a tremendous amount of attention from the scientific community in the past decade [1]. This phenomenon was born as a consequence on initial design consideration undergone in engineering the underlying communication architecture in the automotive domain [2]. Not too long ago, automotive systems were disconnected from the outside world, being self-contained and executed in isolation from the Internet. The advancements in technology and the continuous demand for improved customer services pushed forward the inter-connectivity with external services and infrastructure [3], raising in consequence critical security concerns [4].

Current trends in the automotive field lean towards interconnected and cooperative vehicles, with Vehicle-2-Everything (V2X) [5] being the new *hot-topic*,

only to be accompanied by Connected and Automated Vehicles (CAVs) [6]. On the other end, the sub-systems and communication protocols present in our everyday vehicles were designed with a clear scope and purpose, with the focus not on security features, but rather on *mission-critical* aspects, such as real-time message transmission, network reliability, and tolerance to errors. Modern vehicles incorporate multiple communication systems to satisfy the wide variety of functionalities required by the vehicles and their users.

The domain of security services in the context of automotive systems represents an active research field thoroughly addressed in the past year by individual works and review papers. Authors of [1] offered a comprehensive survey of issues, threats, challenges, and the most relevant solutions targeting the security of in-vehicle systems. More recent works, such as [4], motivate further the need for solid security mechanisms. The authors mention their concern regarding the continuous evolution of vehicle networks (e.g., smart and interconnected vehicles), and the threats that may arise as a consequence of this process. To further address future security concerns in automotive systems, the work from Pham and Xiong [7] explored security threats and existing solutions targeting CAVs specifically. Since automotive systems become more and more connected, the technology is pushed towards inter-vehicle communication and persistent connection with external services, therefore it is difficult to estimate what impact threats will have on system security [8].

The current work takes a step back from the ever evolving technology and intents to assess the formal correctness of two existing solutions proposed in the literature. The first solution, titled Key Distribution Service (KDS) was originally proposed in [9] as a means to distribute long-term encryption and short-term authentication keys between automotive control units. The second solution, titled Mixed data authentication for Controller Area Network (MixCAN), was design to allow control unit to aggregate under a Bloom Filter (BF) [10] multiple authentication tags (e.g., Message Authentication Code (MAC)) that can be verified independently from each other. Later, in [11] these services were brought together under a common system and demonstrated how they can work together to improve system security. While in [11] only an automated formal analysis was conducted leveraging the Schyter [12] verification tool, the work at hand extends the formal analysis with a Burrows–Abadi–Needham (BAN) logic analysis. Consequently, the formal correctness of the two security services is demonstrated from a different analysis point of view.

The paper continues with an overview of the related work in Section 2. Afterwards, Section 3 outlines the security services subjected to the formal analysis. A background of BAN logic is given in Section 4, with the formal analysis conducted in Section 5. The paper concludes in Section 6.

2 Related Work

Considering the context of the paper at hand, the related work outlines relevant studies available in the literature that solve security problems which target

data authentication and key distribution in automotive systems. The considered studies tackle to provide solution to threats and issues as pointed out in [1, 4].

A similar work as the present one, is that of Lauser et al. [13]. In their paper, the authors conducted a formal analysis using the formal verification tool Tamarin [14] of the Secure Onboard Communication (SecOC) [15] standard. SecOC encapsulates a series of recommendation in terms of data authentication for in-vehicle networks developed by the standardisation organisation Automotive Open System Architecture (AUTOSAR) [15]. Mundhenk et al. [16] proposed a framework aiming to allow key exchanges between control unit at system runtime with minimal overhead on control unit resources. Their approach combines symmetric and asymmetric encryption schemes, ensuring in parallel authentication and authorisation for data streams. On the same topic, there is the work of Youn et al. [17]. Here, the authors created an efficient key management scheme with low network overhead, that was validated both, on real hardware and from a formal point of view.

Earlier works [18, 19, 20, 21, 20] in automotive security, similar to [16, 17], follow the same design principals. Their solutions aim to satisfy specific properties of the underlying system. Meaning, their approaches intent to leverage the underlying communication protocol (e.g., Controller Area Network (CAN) [22]) without requiring protocol message changes. Moreover, the protocols envisioned in these works tend to leverage cryptographic operation that would require low computational cost when executed on control units.

3 Security Services

The section at hand aims to briefly outline the main functionalities of the security services considered for formal analysis. Consequently, details in terms of design, method properties and background on algorithms used are omitted. Moreover, the omitted details can be found in the original papers. The initial design of KDS can be found in the work of Genge and Haller [9], and an extended version in [11]. Likewise, the original work of MixCAN is in [23] and its extension in [11].

3.1 Key Distribution Service

The KDS is meant to achieve three distinct functionalities. First, KDS is responsible to distribute cryptographic symmetric long-term encryption keys to a group of protocol members. Similarly, KDS enables dynamic key distribution for short-term authentication keys to the same group of members. Lastly, KDS enables protocol members to synchronise themselves with the group distributor via a key synchronisation mechanism. A given system, may run multiple instances of KDS, as such, for a given instance of KDS there is a set of key receivers and a single key distributor. The KDS is initiated by a protocol initiator i (e.g., key distributor) in relation to a set of receivers r. For each of the functionality mentioned above, KDS builds a separate protocol:

- Long-Term Key Protocol (Proto-LTK): Executed by a key distributor i to share a long-term symmetric encryption key K to a group of members $r \in R$.
- Short-Term Key Protocol (Proto-STK): Executed by a key distributor i to share a short-term symmetric authentication key k to a group of members $r \in R$.
- Symmetric Key Synchronisation Protocol (Proto-SKS): Executed by a group member $r \in R$ to update his encryption or authentication key.

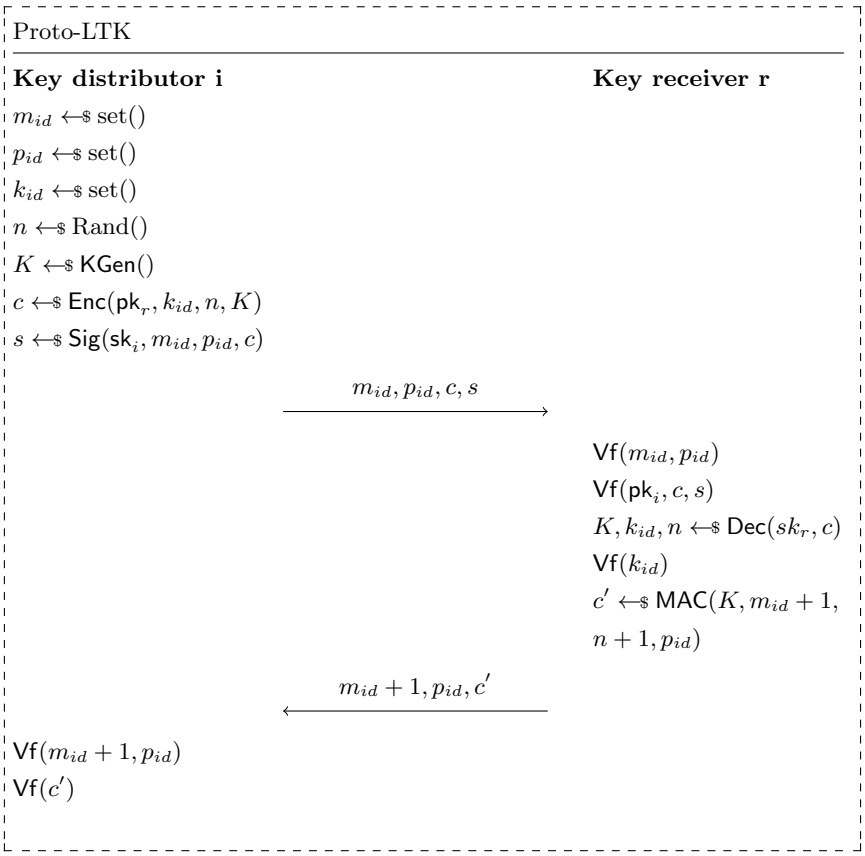

Fig. 1. Proto-LTK message exchange.

Each receiver $r \in R$ is required to know the long-term public key of i, denoted by pk_i, while i is required to know the shared long-term key pk_r by each $r \in R$. The bootstrapping process for a key distributor i entails the generation of a pair of long-term asymmetric keys (pk_i, sk_i). The pk_i is securely distributed to each $r \in R$. Likewise, for each $r \in R$, the new pair of keys (pk_r, sk_r) is securely generated and loaded in $r \in R$, and pk_r is installed in i. To run a round of

Proto-LTK, i broadcasts to each $r \in R$ a sequence of messages consisting in a public part, a private part c, and an authentication part s. If a r manages to verify the sequence received, it responds back with a confirmation and a proof demonstrating the usage of the new key encryption K. If an error occurs, r is required to run Proto-SKS.

In Figure 1 the communication exchange can be viewed. Before initiating the communication with the group R, i executes several steps. First, i sets the protocol message id m_{id}, protocol id p_{id} and key id k_{id} via $set()$ operations. Afterwards, i generates a protocol freshness value n. Subsequently, i executes $KGen()$ to create a new symmetric key (e.g., AES key) K. Afterwards, i proceeds to compute the private message part of Proto-LTK using the bootstrapped master asymmetric key pk_r. This consists in the encryption key K, the freshness value n, and the k_{id}. Lastly, i computes a digital signature with its private key sk_i over the secret message part c, and the public part m_{id} and p_{id}.

In the first interaction, i sends m_{id} and p_{id} as plain text (e.g., public part), accompanied by the encrypted part c and the associated signature s. After receiving these terms, each $r \in R$ follows the same exact steps to update their encryption key K. Each r is required to first perform a sanity check over the public terms m_{id} and p_{id}. Next, r proceed to verify signature s by executing a $Verif()$ operation with i public key pk_i. If s is successfully verified, r decrypts c using its secret sk_r. Once obtaining the encryption key K, r verifies the key identifier k_{id}. If k_{id} is greater by 1 from the previous k_{id}, r computes the acknowledgement message for i. This implies a MAC operation with K to provide proof to i that the new key was received and used successfully. To finalise the protocol Proto-LTK, i is required to check the acknowledgement messages received from r. If correct, the protocol was executed successfully.

The second protocol, Proto-STK is meant to be executed on top of Proto-LTK. Proto-STK leverages the Proto-LTK encryption key K, to distribute a short-term authentication key k. In terms of protocol design, Proto-STK and Proto-LTK follow the same structures and terms, with the difference being in the type of key distributed. Consequently, the roles in Proto-STK are unchanged from Proto-LTK. The protocol Proto-STK is outlined in Figure 2. In Proto-STK the private parts c and c' are computed with key K on both, i and r. The protocol is initiated by i which broadcasts a sequence of terms consisting of m_{id} and p_{id} as public part, a k_{id}, freshness n and new short-term key k as the private part protected by the Proto-LTK key K; and a digital signature computed over both the private and public part. Once obtaining the terms, each r is required to provide the proof of usage for the new key as response.

If one protocol, either Proto-LTK or Proto-STK fails, Proto-SKS offers the possibility of synchronisation. This protocol is an alternative to re-running the whole sequence from the beginning. Proto-SKS is designed to function for both Proto-LTK and Proto-STK, the differentiation between the two being accomplished via the protocol type term p_{type}. Of course, Proto-SKS doesn't cover all the possibilities in error handling. While Proto-SKS is initiated by a protocol member, there should exist a mechanism also that determines if too many

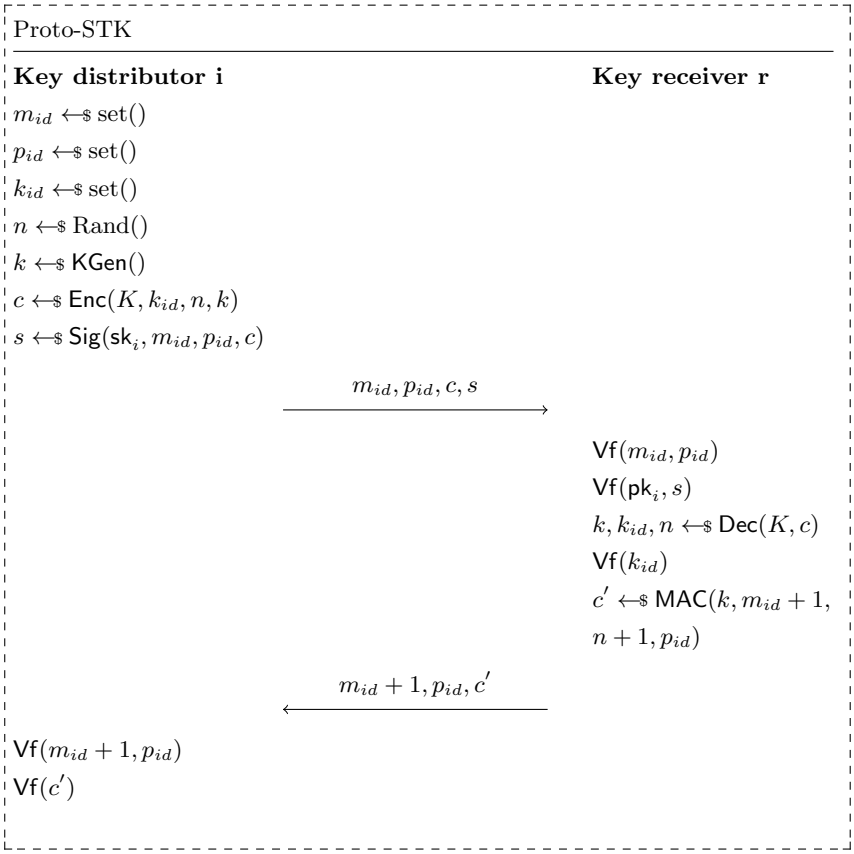

Fig. 2. Proto-STK message exchange.

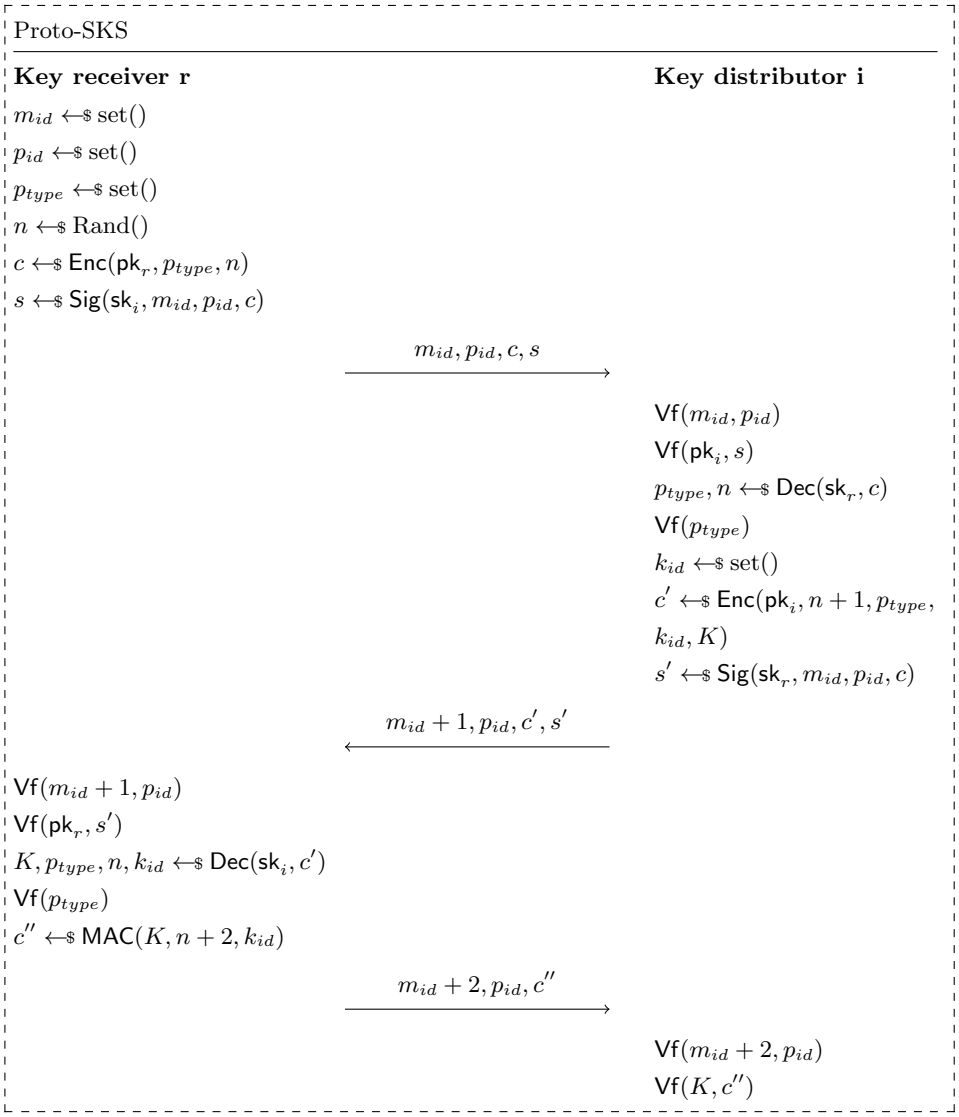

Fig. 3. Proto-SKS message exchange.

members failed to update their keys, which is the responsibility of the protocol distributor. As shown in Figure 3, Proto-SKS is initiated by a protocol receiver r by requesting via a challenge a new key to the distributor i. The challenge consists of a freshness nonce n and the p_{type} signed with the private key of r. If successfully verified, i responses with a new key depending on p_{type}. The term K can denote both, a Proto-LTK or Proto-STK key, in relation with p_{type}.

Consequently, r is required to confirm that the distribution of the new key was successfully with an acknowledge.

3.2 Data Authentication Service

The data authentication approach MixCAN builds on the attributes of the BF [10], allowing a protocol member to authenticate and aggregate messages with different identifiers in a single data structure. A verifier can check the authentication tag for a subset of monitored message identifiers. For this purpose, MixCAN requires the presence of a short-term symmetric authentication cryptographic key k that has been securely distributed amongst all communicating members. This feature can be enabled through security protocols such as ProtoSTK.

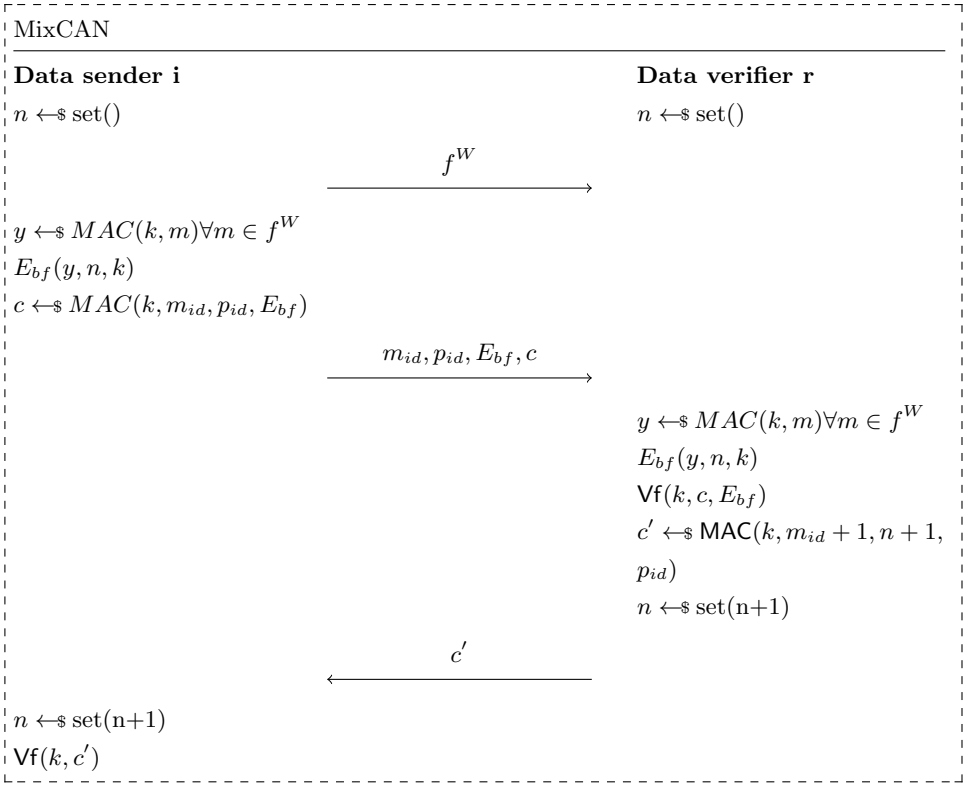

Fig. 4. MixCAN message exchanges.

MixCAN decouples the transmitted data from the actual authentication process. Accordingly, the authentication tag structure is transmitted at a later time on an aggregated and mixed set of messages f according to a time window W.

MixCAN builds on the concept of BFs, and in particular, on the concept of Encrypted Bloom Filter (EBF) [24]. Accordingly, the family of hash functions responsible for inserting an element into the BF data structure are replaced by encryption algorithms. To showcase the workflow of MixCAN, the following roles are defined. Let i be the protocol initiator that periodically sends to a protocol receiver r a sequence of messages f^W over the time window W. After x frame were sent, i proceeds to computes $E_{bf}(f^W, n, k)$. Once the E_{bf} is computed, i computes an additional authentication tag over it (e.g., MAC), and transmits the structure to r. If r successfully verifies the obtained structure, it responds with a confirmation message to i. This procedure is outlined in Figure 4, where m_{id} is the message identifier, p_{id} is the protocol identifier associated with MixCAN.

After having received the sequence of messages f_s^W from i, the recipient r first verifies the validity of the MAC by recomputing it with the same pre-shared cryptographic key k. If successful, then it proceeds with the computation of the MAC for f_s^W, obtaining thus the value y_x. By following similar steps as in the case of the insertion procedure, the verifier then splits y_x into $\lceil log_2(m) \rceil$ parts. As a last step, the verifier checks if all bits identified are in E_{bf}. If so, it concludes that the authentication tag is valid.

4 BAN Logic

BAN logic [25] is designed as a formal logic system utilised for analysing security protocols. Its primary objective is to establish a framework enabling protocol designers to demonstrate the formal correctness of security protocols, abstracting away from specific low-level details. In this current study, the BAN logic is employed to assess the formal correctness of the considered protocols. This logic of authentication involves two epochs in the execution of a protocol: the past and the present. The protocol begins in the present epoch, emphasising the importance of ensuring that past messages do not impact the beliefs held in the present. Likewise, it is crucial to maintain the stability of beliefs in the present epoch, preventing them from being altered by past beliefs. Additionally, BAN logic abstracts away from specific implementation details or encryption schemes, focusing on the higher-level aspects of protocol analysis.

In this authentication logic, the participants of a protocol are referred to as *principals*. Each principal holds certain beliefs about various protocol elements, such as secret keys and freshness nonces. These beliefs are represented as formulas or statements in BAN logic. Table 1 provides the notations for these constructs. In the table, A, B, or S represent the principals involved in the protocol. K_{ab} denotes a shared key between principal A and B, while K_a and K_a^{-1} represent the public and private keys of principal A, respectively. N_a represents a statement, and X and Y are variables that range over statements.

BAN logic defines a series of constructs to express different assumptions and authentication goals about a protocol. The BAN logic construct are outlined in Table 2. Along constructs, the logic leverages *postulates* or deduction rules for

Table 1. BAN logic notation.

Notation	Definition
A, B, I, R	Principals
K_{ab}	Shared key between A and B
K_a	Public key of A
K_a^{-1}	Private key of A
N_a	Statement (e.g., nonce of a)
X, Y	Range over statements

Table 2. BAN logic constructs.

Notation	Definition
$A \mid\equiv X$	A **believes** X
$A \triangleleft X$	A **sees** X
$A \mid\sim X$	A **said** X
$A \Rightarrow X$	A **controls** X
#(X)	X **is fresh**
$A \overset{K}{\longleftrightarrow} B$	A **and** B **share** K
$\langle X \rangle_Y$	X **combines** Y
$\{X\}_K$	X **encrypted with** K
$\overset{k}{\mapsto} B$	K **pubkey of** B

formal proofs. Burrows et al. [25] defines 5 main postulates. The first rule is the *message meaning rule* and is expressed as:

$$\frac{A \mid\equiv A \overset{K}{\longleftrightarrow} B, A \triangleleft \{X\}_K}{A \mid\equiv (B \mid\sim X)}.$$

The above rule can be adapted for public keys as follows:

$$\frac{A \mid\equiv A \overset{K_a}{\longleftrightarrow} B, A \triangleleft \{X\}_{K_a^{-1}}}{A \mid\equiv (B \mid\sim X)}.$$

The message meaning rule can be interpreted as follows. If a principal A believes it shares a secret key K with a principal B, and if A receives a message X encrypted with key K from B, then A is allowed to believe that B once said X. The same rationale applies for the public/private key and shared secret variants of the rule. Next, the *nonce-verification rule* is the second formula denoted as:

$$\frac{A \mid\equiv \#(X), A \mid\equiv (B \mid\sim X)}{A \mid\equiv (B \mid\equiv X)}.$$

Here, the rule states that if a principal A believes that a statement X is fresh, and if A additionally believes that B once said X, then in consequence A believes that B believes X. Thirdly, there is the *jurisdiction rule*:

$$\frac{A \mid\equiv (B \Rightarrow X), A \mid\equiv (B \mid\equiv X)}{A \mid\equiv X}.$$

In the jurisdiction statement, if a principal A believes there is a principal B that has control over X, and B also believes X, then in consequence A will

believe X. Forth postulates concern the fact that if a principal is able to see a formula, then it also sees its components since it knows the necessary keys:

$$\frac{A \triangleleft (Y, X)}{A \triangleleft X}, \quad \frac{A \triangleleft \langle X \rangle_Y}{A \triangleleft X}, \quad \frac{A \mid\equiv B \overset{K}{\longleftrightarrow} A, A \triangleleft \{X\}_K}{A \triangleleft X}, \quad \frac{A \mid\equiv \overset{K}{\longmapsto} B, A \triangleleft \{X\}_K}{A \triangleleft X},$$

$$\frac{A \mid\equiv \overset{K_b}{\longmapsto} B, A \triangleleft \{X\}_{K_b^{-1}}}{A \triangleleft X}.$$

Lastly, a rule is given stating that if a formula is fresh, the entire formula must also be fresh:

$$\frac{A \mid\equiv \#(X)}{A \mid\equiv \#(X, Y)}.$$

5 Analysis

By leveraging the previously outlined BAN logic notations, constructs and postulates, in this section each service is formally analysed. Analysing a protocol with BAN logic implies first the definition of a set of assumption hold about the protocol properties. Afterwards, a set of authentication goals that must hold after the analysis need to be defined. The preliminary steps of the analysis continues with the definition of the protocol messages and its idealised version. In the idealised version, protocol terms must be replaced with BAN logic formals based on the raised assumptions. Lastly, the BAN logic postulates are applied to verify the authentication goals.

5.1 Proto-LTK

Let I and R be the protocol principals, where I denotes the protocol initiator and R the protocol receiver. I is meant to distribute a new symmetric key to R. I and R hold the following assumptions:

Proto-LTK assumptions	
I believes:	R believes:
$R \overset{K_{rp}}{\longleftrightarrow} I$	$I \overset{K_{ip}}{\longleftrightarrow} R$
$I \mid\equiv \overset{K_{rp}}{\longmapsto} R$	$R \mid\equiv \overset{K_{ip}}{\longmapsto} I$
$\#(N+1)$	$\#(N)$
	$\#(k_{id})$
	$I \Rightarrow I \overset{K_{ir}}{\longleftrightarrow} R$
	$I \Rightarrow I \overset{k_{id}}{\longleftrightarrow} R$

Accordingly, the protocol assumes $I \neq R$. From the point of view of I, I believes that its public key K_{ip} is shared with R. Likewise, R believes I knows it's public key K_{rp}. R believes that the nonce N received from I is fresh. In return, I believes that the returned incremented nonce from R is fresh. Both, I and R believe that I has jurisdiction over the symmetric key K_{ir} and its identifier k_{id}. Furthermore, I and R believe that k_{id} is fresh, and was not sent before in the past. With these assumptions, the protocol intents to achieve the following authentication goals:

Proto-LTK authentication goals	
I believes:	R believes:
$I \overset{K_{ir}}{\longleftrightarrow} R$	$I \overset{K_{ir}}{\longleftrightarrow} R$
$R \mid\equiv I \overset{K_{ir}}{\longleftrightarrow} R$	$I \mid\equiv I \overset{K_{ir}}{\longleftrightarrow} R$

The main authentication goals state that after a protocol run, I and R share a new key K_{ir}. Below the Proto-LTK message sequence is offered:

1. $I \rightarrow R : m_{id}, p_{id}, \{k_{id}, N, K_{ir}\}_{K_{rp}}, \{m_{id}, p_{id}, \{k_{id}, N, K_{ir}\}_{K_{rp}}\}_{K_i^{-1}}$
2. $R \rightarrow I : m_{id}+1, p_{id}, \{m_{id}+1, p_{id}, k_{id}, N+1\}_{K_{ir}}$

Next, the idealised messages are given:

1. $I \rightarrow R : \{\#(k_{id}), \#(N), K_{ir}\}_{K_{rp}}, \{\{\#(k_{id}), \#(N), K_{ir}\}\}_{K_i^{-1}}$
2. $R \rightarrow I : \{\#(k_{id}), \#(N+1)\}_{K_{ir}}$

Note that the public message parts m_{id} and p_{id} are omitted from both the public and private part in the idealised version. In BAN logic this is omitted since it represents parts that can be forged by an adversary. The idealised protocol is analysed by applying the BAN logic rules over the raised assumptions.

Message 1 is received by R and $R \triangleleft (M, S)$, where $M = (\{\#(k_{id}), \#(N), K_{ir}\}_{K_{rp}})$ and $S = (\{\{\#(k_{id}), \#(N), K_{ir}\}\}_{K_i^{-1}})$. Because of the following hypothesis:

$$R \mid\equiv \overset{K_{ip}}{\longmapsto} I, \quad I \mid\equiv \overset{K_{rp}}{\longmapsto} R,$$

the message-meaning rule for public keys can be applied on M and S:

$$\frac{R \mid\equiv R \overset{K_{rp}}{\longleftrightarrow} I, R \triangleleft \{M\}_{K_{rp}}}{R \mid\equiv (I \mid\sim M)}, \quad \text{and} \quad \frac{R \mid\equiv I \overset{K_{ip}}{\longleftrightarrow} R, R \triangleleft \{S\}_{K_i^{-1}}}{P \mid\equiv (I \mid\sim S)},$$

yielding

$$R \mid\equiv I \mid\sim (\{\#(k_{id}), \#(N), K_{ir}\}_{K_{rp}}, \{\{\#(k_{id}), \#(N), K_{ir}\}\}_{K_i^{-1}}).$$

Next, the following hypothesis are raised: $R \mid\equiv \#(N), \quad R \mid\equiv \#(k_{id})$. By apply the nonce-verification rule for N and k_{id} it is obtained:

$$\frac{R \mid\equiv \#(N), R \mid\equiv (I \mid\sim N)}{R \mid\equiv (I \mid\equiv N)} \quad \text{and} \quad \frac{R \mid\equiv \#(k_{id}), R \mid\equiv (I \mid\sim k_{id})}{R \mid\equiv (I \mid\equiv k_{id})},$$

the postulate yields $R \mid\equiv I \mid\equiv (N, k_{id})$. Additionally, the freshness rule holds:

$$\frac{R \mid\equiv \#(N)}{R \mid\equiv \#(N, k_{id})}.$$

Subsequently, the hypothesis follows $R \mid\equiv I \Rightarrow K_{ir}, \quad R \mid\equiv I \Rightarrow k_{id}$. The jurisdiction rule applied is applied:

$$\frac{R \mid\equiv (I \Rightarrow K_{ir}), R \mid\equiv (K_{ir})}{R \mid\equiv I \overset{K_{ir}}{\longleftrightarrow} R}.$$

The same can be stated about k_{id}, with the conclusion being $R \mid\equiv R \overset{K_{ir}}{\longleftrightarrow} I, \quad R \mid\equiv R \overset{k_{id}}{\longleftrightarrow} I$.

This concludes the analysis of Message 1. In Message 2, R responds back to I with the incremented received nonce $N + 1$, the key id k_{id}, both encrypted with the new key K_{ir}. This message intents to demonstrate that R managed to extract and use the K_{ir}. Message 2 is received by I and

$$I \triangleleft \{\#(k_{id}), \#(N + 1)\}_{K_{ir}}.$$

The message meaning rule for symmetric keys is applied:

$$\frac{I \mid\equiv K_{ir}, I \triangleleft \{M\}_{K_{ir}}}{I \mid\equiv (R \mid\sim M)},$$

where $M = (\{\#(k_{id}), \#(N+1)\}_{K_{ir}})$. By applying the nonce-verification rule it is obtained:

$$\frac{I \mid\equiv \#(N + 1), I \mid\equiv (R \mid\sim N + 1)}{I \mid\equiv (R \mid\equiv N + 1)},$$

the hypothesis $I \mid\equiv \#(N + 1)$ holds. With the freshness formula:

$$\frac{I \mid\equiv \#(N + 1)}{I \mid\equiv \#(N + 1, k_{id})}.$$

Lastly, the jurisdiction rule is applied:

$$\frac{R \mid\equiv (I \Rightarrow K_{ir}), R \mid\equiv (I \mid\equiv K_{ir})}{R \mid\equiv K_{ir}}.$$

This concludes the analysis of Message 2.

5.2 Proto-STK

The protocol Proto-STK considers the same principals as Proto-LTK, I as a protocol initiator, and R as a protocol responder. While in Proto-LTK I distributes a long-term symmetric key to R, in Proto-STK I leverages the long-term symmetric key to distribute a shot-term authentication key to R. Likewise, let K_i^{-1} denote the private key of I and K_i the public part. For convenience, let $K_{ir}^{'}$ be the new authentication key distributed by I, and K_{ir} the long-term key generated in Proto-LTK. Next, the protocol assumptions are defined:

Proto-STK assumptions	
I believes:	R believes:
$R \overset{K_{ir}}{\longleftrightarrow} I$	$I \overset{K_{ir}}{\longleftrightarrow} R$
$\#(N+1)$	$\#(N)$
	$\#(k_{id})$
	$I \Rightarrow I \overset{k_{id}}{\longleftrightarrow} R$
$\overset{K_{rp}}{\longmapsto} R$	$\overset{K_{ip}}{\longmapsto} I$

Afterwards, the following authentication goals are raised:

Proto-STK authentication goals	
I believes:	R believes:
$I \overset{K_{ir}^{'}}{\longleftrightarrow} R$	$I \overset{K_{ir}^{'}}{\longleftrightarrow} R$
$R \mid\equiv I \overset{K_{ir}^{'}}{\longleftrightarrow} R$	$I \mid\equiv I \overset{K_{ir}^{'}}{\longleftrightarrow} R$

The authentication goals target the successful distribution of the authentication key $K_{ir}^{'}$, to allow R to compute in the future authentication tags. Subsequently, the protocol messages are outlined:

1. $I \rightarrow R : m_{id}, p_{id}, \{k_{id}, N, K_{ir}^{'}\}_{K_{ir}}, \{m_{id}, p_{id}, \{k_{id}, N, K_{ir}^{'}\}_{K_{ir}}\}_{K_i^{-1}}$
2. $R \rightarrow I : m_{id}+1, p_{id}, \{m_{id}+1, p_{id}, k_{id}, N+1\}_{K_{ir}^{'}}$

Likewise, the protocol messages are given in the idealised model:

1. $I \rightarrow R : \{\#(k_{id}), \#(N), K_{ir}^{'}\}_{K_{ir}}, \{\{\#(k_{id}), \#(N), K_{ir}^{'}\}_{K_{ir}}\}_{K_i^{-1}}$
2. $R \rightarrow I : \{\#(k_{id}), \#(N+1)\}_{K_{ir}^{'}}$

For simplicity, only the results of the postulates are given. The analysis begins with Message 1, where $R \triangleleft (M, S)$, with $M = (\{\#(k_{id}), \#(N), K_{ir}^{'}\}_{K_{ir}})$ and $S = (\{\{\#(k_{id}), \#(N), K_{ir}^{'}\}_{K_{ir}}\}_{K_i^{-1}})$. By following the same sequence of applying postulates on Message 1 as in the Proto-LTK analysis, we obtain the followings outcomes: $R \mid\equiv I \mid\sim (M, S)$, $R \mid\equiv I \mid\equiv (N, k_{id}, K_{ir}^{'})$, $R \mid\equiv \#((N, K_{ir}^{'}))$, and $R \mid\equiv I \overset{K_{ir}^{'}}{\longleftrightarrow} R$. The same outcomes hold for Message 2.

5.3 Proto-SKS

The Proto-SKS protocol is initiated by R in order to request a new long-term or short-term key from I. Let K_{ir} denote the cryptographic key requested by R based on the protocol type p_{type}. For the protocol assumptions, Proto-SKS follows the same assumption and authentication goals as Proto-LTK. Next, the protocol messages are defined:

1. $R \rightarrow I : m_{id}, p_{id}, \{p_{type}, N\}_{K_{ip}}, \{m_{id}, p_{id}, \{p_{type}, N\}_{K_{ip}}\}_{K_r^{-1}}$
2. $I \rightarrow R : m_{id}+1, p_{id}, \{N+1, p_{type}, k_{id}, K_{ir}\}_{K_{ip}}, \{m_{id}+1, p_{id}, \{N+1, p_{type}, k_{id}, K_{ir}\}_{K_{ip}}\}_{K_i^{-1}}$
3. $R \rightarrow I : m_{id}+2, \{N+2, p_{id}\}_{K_{ir}}$

with the associated idealised protocol:

1. $R \rightarrow I : \{\#(N)\}_{K_{ip}}, \{\{\#(N)\}_{K_{ip}}\}_{K_r^{-1}}$
2. $I \rightarrow R : \{\#(N+1), K_{ir}\}_{K_{ip}}, \{\{\#(N+1), K_{ir}\}_{K_{ip}}\}_{K_i^{-1}}$
3. $R \rightarrow I : \{\#(N+2)\}_{K_{ir}}$

Following the same analysis methodology, for Message 1 the postulates state that: $I \mid\equiv R \mid\sim (M, S)$, where $M = (\{\#(N)\}_{K_{ip}})$ and $S = (\{\{\#(N)\}_{K_{ip}}\}_{K_r^{-1}})$ since $I \mid\equiv R \overset{K_{rp}}{\longmapsto} I$, $I \mid\equiv R \mid\equiv N$, and $I \mid\equiv \#(N)$. The analysis on Message 2, since $R \mid\equiv \overset{K_{ip}}{\longmapsto} I$ yields $R \mid\equiv I \mid\sim (M, S)$, $R \mid\equiv \#((N, k_{id}, K_{ir}))$, and $R \mid\equiv I \overset{K_{ir}}{\longleftrightarrow} R$. Lastly, the Message 3 analysis yields $I \mid\equiv \#(M)$.

5.4 MixCAN

In MixCAN, I and R represent two principals that want to exchange authenticated messages. I and R share a short-term symmetric authentication key K_{ir}, which was obtain through Proto-STK. Consequently, the following assumptions are defined:

MixCAN assumptions	
I believes:	R believes:
$R \xleftrightarrow{K_{ir}} I$	$I \xleftrightarrow{K_{ir}} R$
$R \xleftrightarrow{N} I$	$I \xleftrightarrow{N} R$
$\#(N)$	$\#(N)$
$\#(N+1)$	

with the authentication goals stating that at the end of a protocol round, R believes that E_{bf} is fresh:

MixCAN authentication goals	
I believes:	R believes:
	$\#(E_{bf})$

Next, the MixCAN protocol messages are defined:

1. $I \rightarrow R : f_s$
2. $I \rightarrow R : m_{id}, p_{id}, E_{bf}, \{m_{id}, p_{id}, E_{bf}, N\}_{K_{ir}}$
3. $R \rightarrow I : \{m_{id}, N+1, p_{id}\}_{K_{ir}}$.

Likewise, the idealised messages are outlined:

2. $I \rightarrow R : E_{bf}, \{E_{bf}, I \xleftrightarrow{\#(N)} R\}_{K_{ir}}$
3. $R \rightarrow I : \{\#(N+1)\}_{K_{ir}}$

The analysis begins with Message 2, where $R \triangleleft (E_{bf}, \{E_{bf}, I \xleftrightarrow{\#(N)} R\}_{K_{ir}})$. The hypotheses holds because $R \mid\equiv R \xleftrightarrow{K_{ir}} I$. By applying the message meaning rule for symmetric keys it is obtained that $R \mid\equiv I \mid\sim (E_{bf}, \{E_{bf}, I \xleftrightarrow{\#(N)} R\}_{K_{ir}})$. By applying the nonce-verification rule, it is obtained that $R \mid\equiv I \mid\equiv \#(N)$, and since N is part of $\{E_{bf}, I \xleftrightarrow{\#(N)} R\}_{K_{ir}}$, from the freshness formula it can be stated that $R \mid\equiv \#(E_{bf})$. Continuing with the analysis of Message 3, $I \triangleleft \{\#(N+1)\}_{K_{ir}}$. Similarly, the hypotheses holds since $I \mid\equiv R \xleftrightarrow{K_{ir}} I$. The message meaning rule for symmetric keys results in $R \mid\equiv I \mid\sim \{\#(N+1)\}_{K_{ir}}$. The nonce-verification rule gives $I \mid\equiv R \mid\equiv \#(N+1)$, resulting in $I \mid\equiv \#(\{\#(N+1)\}_{K_{ir}})$. This concludes the analysis of MixCAN.

6 Conclusion

The paper at hand intended to extend existing work on two security services design for automotive systems through a BAN logic analysis. While the original works target the initial design and experimental assessment of the considered security solutions, namely KDS and MixCAN, the formal validation was limited to an automated formal verification with the Scyther modelling checker. The present work goes a step forward in this direction and applies BAN logic to formally check the correctness of the security services.

Acknowledgment

This work is supported by the OPEVA project that has received funding within the Key Digital Technologies Joint Undertaking (KDT JU) from the European Union's Horizon Europe Programme under grant agreement No 101097267 and was co-funded by the Swiss State Secretariat for Education, Research and Innovation (SERI) and the Innosuisse – Swiss Innovation Agency. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or KDT JU. Neither the European Union nor the granting authority can be held responsible for them.

Bibliography

[1] A. Martínez-Cruz, K. A. Ramírez-Gutiérrez, C. Feregrino-Uribe, and A. Morales-Reyes, "Security on in-vehicle communication protocols: Issues, challenges, and future research directions," *Computer Communications*, vol. 180, pp. 1–20, 12 2021.

[2] I. Pekaric, C. Sauerwein, S. Haselwanter, and M. Felderer, "A taxonomy of attack mechanisms in the automotive domain," *Computer Standards & Interfaces*, vol. 78, p. 103539, 2021.

[3] T. Snyder and G. Byrd, "The Internet of Everything," *Computer*, vol. 50, no. 6, pp. 8–9, 2017.

[4] R. S. Rathore, C. Hewage, O. Kaiwartya, and J. Lloret, "In-Vehicle Communication Cyber Security: Challenges and Solutions," *Sensors*, vol. 22, no. 17, p. 6679, 9 2022.

[5] A. Alalewi, I. Dayoub, and S. Cherkaoui, "On 5G-V2X Use Cases and Enabling Technologies: A Comprehensive Survey," *IEEE Access*, vol. 9, pp. 107 710–107 737, 2021.

[6] A. Taeihagh and H. S. M. Lim, "Governing autonomous vehicles: emerging responses for safety, liability, privacy, cybersecurity, and industry risks," *Transport Reviews*, vol. 39, no. 1, pp. 103–128, 1 2019.

[7] M. Pham and K. Xiong, "A survey on security attacks and defense techniques for connected and autonomous vehicles," *Computers & Security*, vol. 109, p. 102269, 10 2021.

[8] A. Nanda, D. Puthal, J. J. P. C. Rodrigues, and S. A. Kozlov, "Internet of Autonomous Vehicles Communications Security: Overview, Issues, and Directions," *IEEE Wireless Communications*, vol. 26, no. 4, pp. 60–65, 8 2019.

[9] B. Genge and P. Haller, "Cryptographic Key Distribution Protocol with Trusted Platform Module for Securing In-vehicle Communications," 2022, pp. 796–807.

[10] B. H. Bloom, "Space/Time Trade-offs in Hash Coding with Allowable Errors," *Commun. ACM*, vol. 13, no. 7, pp. 422–426, 7 1970. [Online]. Available: http://doi.acm.org/10.1145/362686.362692

[11] T. Lenard, B. Genge, P. Haller, A. Collen, and N. A. Nijdam, "An Automotive Reference Testbed with Trusted Security Services," *Electronics*, vol. 12, no. 4, p. 888, 2 2023.

[12] C. J. F. Cremers, "The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols," in *Computer Aided Verification*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 414–418.

[13] T. Lauser, D. Zelle, and C. Krauß, "Security Analysis of Automotive Protocols," in *Computer Science in Cars Symposium*. New York, NY, USA: ACM, 12 2020, pp. 1–12.

[14] S. Meier, B. Schmidt, C. Cremers, and D. Basin, "The TAMARIN Prover for the Symbolic Analysis of Security Protocols," 2013, pp. 696–701.

[15] AUTOSAR, "Specification of Secure Onboard Communication AUTOSAR CP Release 4.3.1," 2017.

[16] P. Mundhenk, S. Steinhorst, M. Lukasiewycz, S. A. Fahmy, and S. Chakraborty, "Lightweight Authentication for Secure Automotive Networks," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015*. New Jersey: IEEE Conference Publications, 2015, pp. 285–288.

[17] T.-Y. Youn, Y. Lee, and S. Woo, "Practical Sender Authentication Scheme for In-Vehicle CAN With Efficient Key Management," *IEEE Access*, vol. 8, pp. 86 836–86 849, 2020.

[18] A. Van Herrewege, D. Singelee, and I. Verbauwhede, "CANAuth - A Simple, Backward Compatible Broadcast Authentication Protocol for CAN bus," in *ECRYPT Workshop on Lightweight Cryptography 2011*, ser. ECRYPT '11, 2011, pp. 1–7.

[19] B. Groza, S. Murvay, A. V. Herrewege, and I. Verbauwhede, "LiBrA-CAN," *ACM Transactions on Embedded Computing Systems*, vol. 16, no. 3, pp. 1–28, 7 2017.

[20] A.-I. Radu and F. D. Garcia, "LeiA: A Lightweight Authentication Protocol for CAN," 2016, pp. 283–300.

[21] S. Woo, H. J. Jo, I. S. Kim, and D. H. Lee, "A Practical Security Architecture for In-Vehicle CAN-FD," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 8, pp. 2248–2261, 8 2016.

[22] ISO, "ISO 11898-1:2003 - Road vehicles - Controller area network (CAN) - Part 1: Data link layer and physical signalling," *International Organization for Standardization*, 2003.

[23] T. Lenard, R. Bolboacă, B. Genge, and P. Haller, "MixCAN: Mixed and Backward-Compatible Data Authentication Scheme for Controller Area Networks," in *2020 IFIP Networking Conference (Networking)*, 2020, pp. 395–403.

[24] S. M. Bellovin and W. R. Cheswick, "Privacy-Enhanced Searches Using Encrypted Bloom Filters," 2004. [Online]. Available: http://eprint.iacr.org/2004/022

[25] M. Burrows, M. Abadi, and R. Needham, "A logic of authentication," *ACM Transactions on Computer Systems*, vol. 8, no. 1, pp. 18–36, 2 1990.