Performance Evaluation of an Authentication Mechanism in Time-Triggered Networked Control Systems

Goncalo Martins, Anirban Bhattacharjee, Abhishek Dubey, and Xenofon D. Koutsoukos

Institute for Software Integrated Systems (ISIS)
Department of Electrical Engineering and Computer Science
Vanderbilt University, Nashville, Tennessee, USA
<firstname.lastname>@vanderbilt.edu

***Abstract*—An important challenge in networked control systems is to ensure the confidentiality and integrity of the message in order to secure the communication and prevent attackers or intruders from compromising the system. However, security mechanisms may jeopardize the temporal behavior of the network data communication because of the computation and communication overhead. In this paper, we study the effect of adding Hash Based Message Authentication (HMAC) to a time-triggered networked control system. Time Triggered Architectures (TTAs) provide a deterministic and predictable timing behavior that is used to ensure safety, reliability and fault tolerance properties. The paper analyzes the computation and communication overhead of adding HMAC and the impact on the performance of the time-triggered network. Experimental validation and performance evaluation results using a TTEthernet network are also presented.**

Keywords*—*Time-Trigger Architectures, TTEthernet, HMAC, Secure Messages, Performance Evaluation

I. Introduction

A system's inability to provide its specified services in the domain of safety-critical control applications such as by-wire systems in automotive systems can threaten human lives. For those ultra-dependable systems, the failure rates for those systems should be in the order of 10^{-9} failures/hour [1]. Cyber-Physical Systems require distributed architectures to support safety critical real-time control. For such systems, Time-Triggered Architectures (TTA) offer signficant advantages in terms of safety, reliability, and fault tolerance [2].

TTA provides a framework to design distributed, embedded, and real-time systems ensuring high-dependability. TTA provides mechanisms to determine precise timing and helps to engineer fault-tolerance systems for both control software and networked data communications [3]. Time-triggered networks are beneficial in many Cyber-Physical System (CPS) applications including safety-critical systems, especially in-vehicle networks (e.g. TTEthernet), by managing the complexity of fault-tolerance and analytic dependability models and ensuring highly reliable and deterministic systems [3].

TTA systems have focused on safety, reliability and fault-tolerance properties. Another important property is to ensure communication security [4, 5]. Providing mechanisms to enable secure communications is important to prevent actions by attackers or intruders. Adding security mechanisms may incur significant overhead and jeopardize the network data communication in such highly dependable timing systems.

In this paper, in order to provide secure communication, an authentication mechanism based on Hash Based Message Authentication (HMAC) for communication is implemented and evaluated in time-triggered network. The paper analyzes the computation and communication overhead of adding HMAC and the impact on the performance of the time-triggered network. A comprehensive evaluation of the computational and network overhead due to message authentication is also presented. Our results can be used for evaluating the performance impact of security mechanisms, and thus, for design of secure time-triggered networked control systems.

The paper is organized as follows: Section I presents the introduction and related work; Section II describes the problem; Section III introduces the model by describing the network topology and the end nodes security. It also presents the results for the computation overhead; Section IV presents the theoretical analysis for the network communications. Section V presents the results for the communication overhead; Section VI concludes this work.

A. Related Work

In [6], an experiment was conducted to evaluate the fragility of the underlying system structure of a modern automobile system. The paper shows how an attacker is able to infiltrate virtually any Electronic Control Unit (ECU). This can leverage the ability to completely circumvent a broad array of safety critical systems, and manipulate and control the automotive functions by ignoring the driver's input. In [7], the National Highway Traffic Safety Administration (NHTSA) reported some cases about infiltration at car's control systems and install malware remotely, using Bluetooth devices and CD. Such vulnerabilities highlight the importance in securing messages in vehicular networks.

As the software complexity increases, mainly due to infotainment (information-based media content or programming), it is almost impossible to avoid security flaws. Drive-by-wire systems can be developed based on time-triggered architectures such as TTEthernet [8]. Security threats relevant to TTEthernet are discussed and classified in 3 main categories: location, cryptographic knowledge, and operational, in [9].

978-1-4799-4187-2/14/$31.00

In [10], a study of current and future bus systems is presented with respect to their security features. The paper states that "a fundamental step to improve automotive bus communication security is the encryption of all vehicular data transmission". A secure automotive communication based on modern cryptographic mechanisms is proposed, however, the performance impact of such security mechanisms is not evaluated.

II. Problem

The problem addressed in this paper is to evaluate the performance impact of adding an authentication mechanism that ensures secure communications in a time-triggered networked control system and analyze how that extra information can jeopardize the computational and network performance. The example is based on an automotive communication system described in [11]. The bus communication system used is TTEthernet[1]. The system is composed by 3 Electronic Control Units (ECUs), 1 TTEthernet switch, and 1 real-time target computer which is used to simulate the vehicle dynamics.

The first goal of the paper is to measure the computational overhead due to HMAC. The implemented method guarantees that the messages are coming from an authenticated node. Every message in the network bus can be authenticated at every node. The entity authentication of sender's message can be verified at the receiver's end and if the sender's message is not authenticated the receiver node will discard the message (Section III). The second goal is to measure the network overhead on the bus communications (Sections IV and V).

III. Model

As mentioned before, the considered experiment platform comprises of a network architecture based on a TTA network. This network allows synchronization among local node's clocks in a centralized way. The nodes are connected through a central switch via bidirectional communication links. Each node communicates with the other nodes by sending messages to the switch, which then relays the messages to the respective receiving nodes.

In TTA systems, events occur at a predefined time with a precision at the microsecond level. The system provides an off-line scheduling tool that statically creates the bus communication schedule table. This table specifies when messages (e.g. time-trigger (TT) or best effort (BE) messages) are transmitted by which node and who will receive the message. This feature ensures that the network gives priority to TT messages without collisions. This approach ensures predictable deterministic behavior, where messages that are not delivered can be easily detected.

A. Network Topology

The ECUs are self-contained units that include a processor with memory, a real time operating system based on RTLinux and Ubuntu (Linux kernel 2.6.24-24-rt) and the respective networking devices drivers to enable time trigger communications [11]. The network is a star topology (Figure 1). All the software developed is running at the kernel space managed by RTLinux scheduler, guaranteeing real time execution. In conjunction with the TTA system scheduler, for every cycle, the created tasks repeat the same schedule enabling the nodes to communicate in a deterministic way. Representative parameters used for the network, such as the cycle or base period, channels bandwidth, clock synchronization cycle, and synchronization precision, can be found in [11].

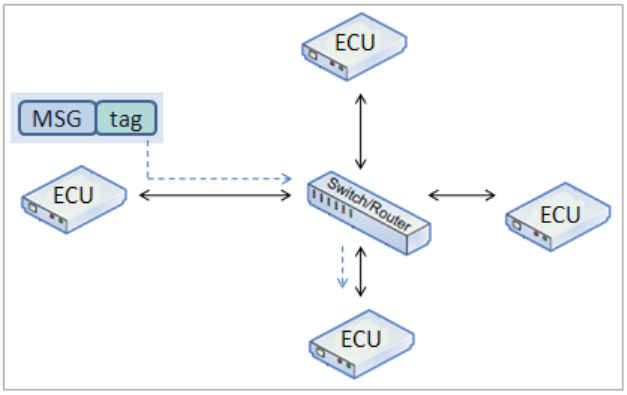

Fig. 1. Network Topology - Star Configuration

B. End Nodes Security

The communication between the nodes can be secured by ensuring that the message integrity and authenticity are not compromised. The nodes should be authenticated and the integrity of the messages of the sender's node should be maintained. The messages should not be tampered, that is, an attacker should not be able to modify the messages remotely and the receiver node should only accept messages from authenticated nodes, discarding the other messages.

The technique implemented to authenticate the messages relies on a keyed-hash message authentication code (HMAC [12]). The idea is to generate a tag for the respective message, and then, the tag is appended and transmitted with the original message (Figure 1).

C. HMAC

HMAC generates a tag by combining a cryptographic hash function with a secret cryptographic key. The cryptographic hash-functions should be *one-way* and *collision resistant*. It is computationally unfeasible to find a message which corresponds to a given message digest, or to find two different messages which produce the same message digest. Any change to a message in transit will, with very high probability, result in a different message digest, and the signature will fail to verify. The strength of the HMAC depends upon the cryptographic strength of the underlying hash function, the size of its hash output, and on the size and quality of the key [12].

In this paper, three types of cryptographic hash functions are implemented:
SHA-1: a 160-bit hash function;
SHA-2: SHA-256 hash function with 32-bit words;
SHA-3: Keccak hash function that supports the same hash lengths as SHA-2, but its internal structure is significantly different from the rest of the SHA family [13].

[1] http://www.tttech.com/technologies/ttethernet/

For all the hash functions described above, a secret cryptographic key with 64 bytes is used. The unique tag message authentication code generated by the hash-algorithms simultaneously verify the date integrity and the authentication of a message. The same key is present in the sender and receiver nodes.

Figure 2 shows the message authentication and validation schematic. The process is outlined below:

- the sender node generates a tag for the desired message to transmit (hash algorithms used);
- the message is appended with the tag and the combination is transmitted to the receiver node;
- the receiver node separates the message and the tag;
- then using the same key as the sender, the receiver node regenerates the tag for the message received;
- the receiver then compares the regenerated tag with the received tag.

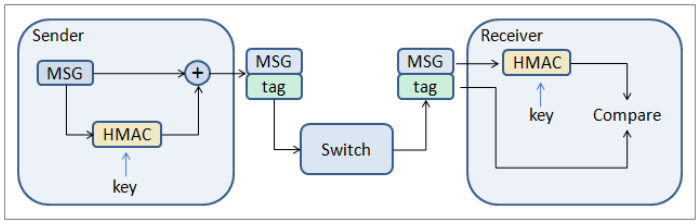

Fig. 2. Message authentication and validation schematic

D. HMAC Performance

The HMAC model from Figure 2 is implemented at the kernel level on the end nodes side in order to guarantee deterministic and fast execution. Each node is an IBX-530W box with an Intel Atom processor running at 1.6 GHz, 1 GB of memory and 512 MB of cache.

Figure 3 presents the minimum, maximum and average execution time for each hash function (SHA1, SHA2 and SHA3). The results show the computational overhead to the execution of the controller code described in [11] is small.

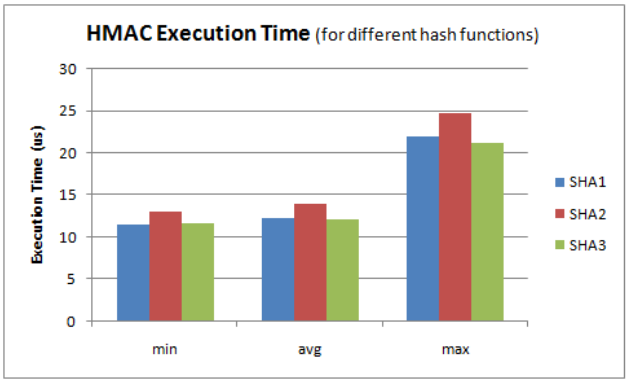

Fig. 3. HMAC Kernel Execution Time for different hash functions

The extra message tag overhead in bytes introduced by enabling secure communications is dependent on the message tag generated by the cryptographic hash-function used in HMAC. In this case, for SHA-1 the message tag is 20 bytes and for SHA-2 and SHA-3 the message tag is 32 bytes.

IV. Analysis

Time-triggered systems use a time-division multiplexing scheme (TDMA) to allocate each message a unique access time within a periodic transmission schedule [2]. Based on this technique, the need for an explicit collision-resolution mechanism is eliminated. Each transmitter determines its turn to access the network by checking a time reference. During the design of the communication system the maximum number of nodes that can participate in the TDMA scheduling should be taken into account. Adding an extra node might disturb the correct operation of the already integrated ones (Figure 4) [2].

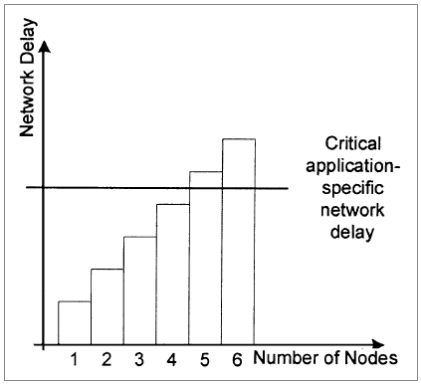

Fig. 4. Maximum network delay at critical instant as a function of the number of nodes [2]

A similar problem is expected to occur if nodes send more information than the maximum allocated per slot. This extra information might be necessary to ensure security during communications. Therefore, it is important to reserve adequate bandwidth to ensure schedulability. This section presents the theoretical analysis of the performance impact on the maximum number of frames by incorporating a tag that changes the frame size.

A. TTEthernet TDMA

Figure 5 shows a typical TTA TDMA frame. It consists of a base period (BP), minimum time between two action times also called guard period (GP) and the frame or slot time ($Frame_{time}$).

The maximum number of frames (NF_{max}) allowed per BP is the following:

$$NF_{max} = \frac{BP}{(Frame_{time})+GP}$$

where, $Frame_{time} = \frac{Packet_{size}}{Transmission_{rate}}$

For all the theoretical and practical results a 100 Mbits/s TTEthernet switch is used. The system defines a minimum frame size of 60 bytes and a maximum frame size of 1514 bytes. It defines also a minimum transmission time between two packets of 0.2 ms (GP = 0.2 ms). This minimum guard period is used to guarantee that the physical switch has time to route one frame size at its maximum size.

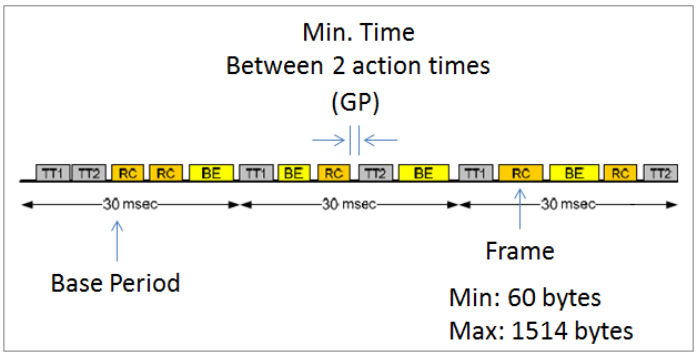

Fig. 5. TTA TDMA Frame

B. Theoretical Results

The base period (BP) is the same as the hyper period from [11], BP = 10 ms. The BP is selected to ensure, that all the control units will work at the designed and desired rate with the proposed secure mechanism implemented. The maximum packet size used in [11] was 60 bytes. As mentioned in Section III the overhead on the frame size due to the generated hash tag is 20 or 32 bytes.

Table I shows the theoretical results for NF_{max} and $Frame_{time}$ with BP = 10 ms and changing the frame size from 60 to 80 bytes. It is also assumed that communications are perfect and fast, i.e., GP = 0. The table also shows the theoretical results for NF_{max} and $Frame_{time}$ with BP = 10 ms and the suggested and defined TTEthernet guard period (GP = 0.2 ms).

TABLE I. THEORETICAL RESULTS [@BP = 10 MS]

	GP = 0		GP = 0.2 ms	
	60 bytes	80 bytes	60 bytes	80 bytes
NF_{max}	208333	156250	48	48
$Frame_{time}$ (ms)	0.0048	0.0064	0.2048	0.2064

Figure 6 shows the expected impact for NF_{max} per BP by changing the frame size from 60 to 1514 bytes. Figure 7 shows the expected impact for $Frame_{time}$ per BP by changing again the frame size from 60 to 1514 bytes.

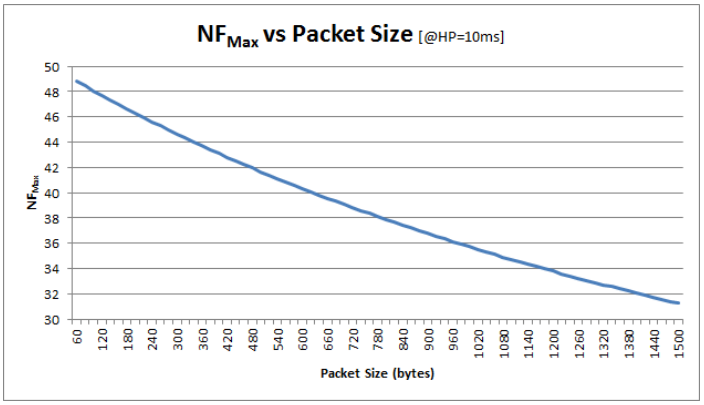

Fig. 6. NF_{max} vs Packet Size (bytes) with BP = 10 ms and GP = 0.2 ms

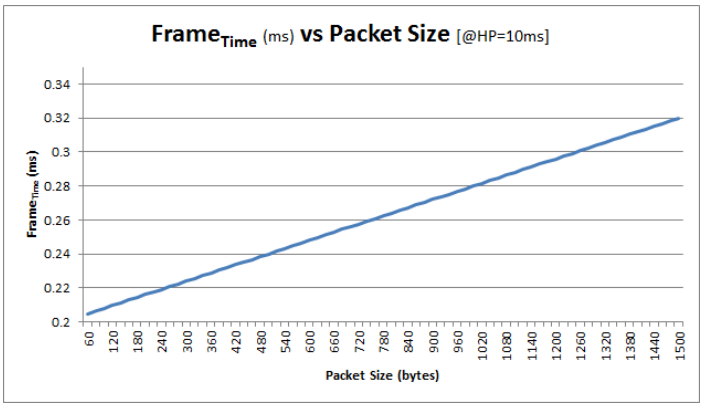

Fig. 7. $Frame_{time}$ (ms) vs Packet Size (bytes) with BP = 10 ms and GP = 0.2 ms

C. Comments

As it can be seen from Table I, by adding a guard period, the maximum number of frames per BP reduces drastically. As an example, for the selected guard period and using the message tag overhead introduced by using SHA-1, there is no impact (in theory) on the maximum number of frames by changing the packet size from 60 to 80 bytes (20 bytes message tag overhead).

V. EXPERIMENTS

This section presents the experimental analysis of the performance impact for the maximum number of frames per BP by changing the amount of data transmitted. For all conducted tests and to better understand the measured results, only two end nodes are used. The results would be the same if all the 4 nodes presented in Section II are included.

A. Physical Measurement Setup

The block diagram for the physical setup is depicted in Figure 8. The goal is to measure the time that a packet takes to travel from one end node to the other through a normal network channel. It is desired to measure this timing with as small interference as possible during packet transmission. With this in mind, it is not feasible to add extra code on the end node side because this will interfere with the real time execution. The solution relies on ”sniffing” the packets on the central switch.

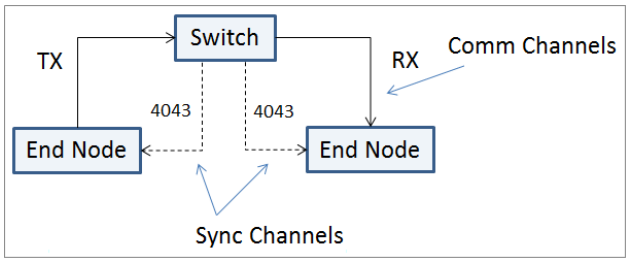

Fig. 8. Block Diagram for the Physical Setup

The central switch is not configured to broadcast the packets to all ports. In order to monitor the packets during communications, it is necessary to connect a secondary switch

that allows the connection of a packet analyzer (e.g. Wireshark[2]). Figure 9 shows the block diagram for the measurement setup. The additional switch (WS) introduces a small delay in the packets transmission time but this delay is consistent and deterministic, as it can be seen in Table II.

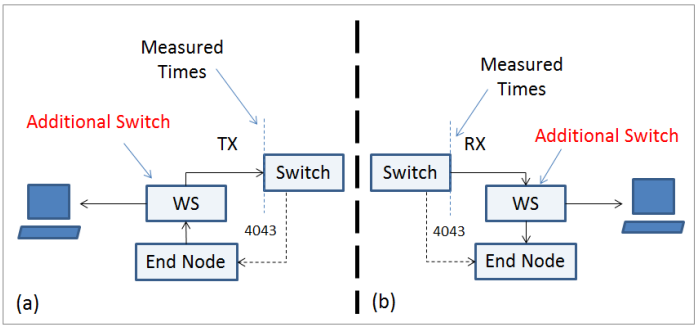

Fig. 9. Block Diagram for the Measurement Physical Setup: (a) Packet Arrival at the Central Switch (b) Packet Transmission at the Central Switch

The central switch is the master node of the TDMA scheduler. It sends to all nodes a sync beacon at a predefined and deterministic time in a dedicated channel (channel 4043). This is the reference packet time used for all measurements. Then the packet arrival time (Tx) and the packet transmission time (Rx) at the switch is measured. With these measurements in conjunction with the sync beacon[3] packet timing, it is possible to infer the packet transmission time from the end node to the switch.

B. Experimental Results

Wireshark is used as the packet analyzer. On the software, it is possible to see the sync beacon packets, the packet arrival and packet transmission timings. All these measurements are related with the central switch. Four tests were conducted in order to evaluate the switch performance at its minimum and maximum frame size. For all the tests, data is collected for 5 minutes. The tests are the following:

- measure the packet arrival time (Tx) for the minimum TTA frame size (60 bytes),
- measure the packet transmission time (Rx) for the minimum TTA frame size (60 bytes),
- measure the packet arrival time (Tx) for the maximum TTA frame size (1514 bytes),
- measure the packet transmission time (Rx) for the maximum TTA frame size (1514 bytes).

Table II shows the values of the performed tests (due to network communication overhead). Based on this data, it is possible to compute the switch performance. For that, it is necessary to compute the time difference (Diff) between the time that the switch received the packet (Tx) with the time that the switch forward the packet (Rx) to the respective channel.

Switch Performance: $Diff = Tx - Rx$

[2]http://www.wireshark.org/

[3]Sync Beacons are frames used as time reference

TABLE II. CENTRAL SWITCH PERFORMANCE MEASURED VALUES

	60 bytes			1514 bytes		
	Min	Avg	Max	Min	Avg	Max
Tx (ms)	0	0.0080	0.1150	0.1100	0.2220	0.3470
Rx (ms)	0.2740	0.3860	0.6730	0.4940	0.5940	0.8260
Diff (ms)	0.2740	0.3780	0.5580	0.3840	0.3720	0.4790

The average value for Diff at 60 bytes (0.378 ms) and 1514 bytes (0.372 ms) is approximately the same and it reflects the consistency and determinism of the central switch in forwarding different frame sizes. In theory this Diff should be the same as the guard period (0.2 ms) mentioned in Section IV. One of the reasons that might contribute for the extra overhead is the addition of the extra switch to sniff the communication channel. It is important to mention that all the values from Table II include this overhead. The additional switch takes different times to forward the packets, that are also dependent on the size of the packets. For the remaining calculations a boundary of 0.1 ms is assumed for the additional switch overhead (SO).

Based on the collected data, it is also possible to get the time that the transmitted packet takes from one end node to the central switch. Once again, this is possible to compute because all end nodes are in sync with the central switch by the sync beacon packet. From Table II, the average transmission time (Tx) for 60 bytes is 0.008 ms and 0.222 ms for 1514 bytes. As expected, the end node takes more time to transmit more data. This is due to the time that the network device drivers at the end node take to make the data available to transmit. Once again, the transmission is not deterministic, it varies with the packet size.

The maximum transmission time (Max_{TT}) can be calculated with the following equation:

$$Max_{TT} = (2 * (Frame_{time} - SO) + Diff)$$

where, $Frame_{time}$ is the time that it takes for the message to go from the end node to the central switch and Diff the time that the central switch takes to forward the message. To simplify the calculations, it is assumed that the network device drivers at the receiver end node take the same time as the network drivers at the transmitter node.

As an example, from Table II, taking the maximum transmission time for 60 bytes (0.115 ms) and with Diff equal to the guard period (0.2 ms), the following Max_{TT} is obtained:

$$Max_{TT} = (2 * (0.115 - 0.1)) + 0.2 = 0.23ms$$

Using SHA-1 message tag overhead and using the equation mentioned above (Max_{TT}), it is possible to evaluate the performance impact on the maximum number of frames per BP by adding extra bytes to ensure secure communications. Table III shows the performance impact values.

TABLE III. PERFORMANCE IMPACT RESULTS

	Theoretical		Measured	
	60 bytes	80 bytes	60 bytes	80 bytes
NF_{max}	48	48	43	33
$Frame_{time}$ (ms)	0.0048	0.0064	0.1150	0.1500
Max_{TT} (ms)	0.2096	0.2128	0.2300	0.3000

Note that the $Frame_{time}$ for the Measured results from Table III already has the additional switch overhead subtracted.

C. Comments

There is a significant drop on the maximum number of frames per BP between the theoretical and measured values. The main reason why is because the theoretical values do not include the time that the device drivers from the end node side take to make available the frame to transmit. The end node network device driver's time is not deterministic and it depends on several factors such as the type of operating system running on it and the amount of data desired to transmit.

However, for the measured values for the application example, there is a small impact on the maximum number of frames per BP by increasing the packet size from 60 to 80 bytes (in this case using SHA-1 message tag). The time that it takes to transmit a frame with or without the generated hash tag is approximately the same and it does not affect the real-time execution of the application example.

VI. Conclusion

Time-triggered networked control systems enable safe, reliable and fault-tolerant network communications. However, it is also important to incorporate secure communications but security mechanisms can affect the overall system stability. Herewith, it is important to evaluate impact performance of secure messages in existing network communications.

In this paper, an authentication method based on a keyed-hashed authentication code (HMAC) is used. The proposed security mechanism enables secure communications between several nodes. The impact on the performance of the bus communication by adding this level of security on the messages is analyzed. The HMAC execution time does not affect the overall system performance. The algorithm is implemented at an operating system kernel level and the execution time is negligible for the nodes that are authenticating the messages.

On the network side, there is a small impact on the maximum number of frames per base period (BP) by changing the number of bytes transmitted. However, despite the need for adding extra bytes in the transmitted packet, the time that it takes to transmit a frame with or without encryption is approximately the same and it does not affect real-time execution.

VII. Acknowledgments

This work is supported in part by the National Science Foundation (CNS-1238959, CNS-1035655) and NIST (70NANB13H169).

References

[1] N. Suri, C. J. Walter, and M. M. Hugue, *Advances in ultra-dependable distributed systems*. IEEE Computer Society Press, 1994.

[2] H. Kopetz and G. Bauer, "The time-triggered architecture," *Proceedings of the IEEE, Vol. 91(1), pp. 112-126*, 2003.

[3] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, "Trends in automotive communication systems," *Proceedings of the IEEE, Vol. 93(6), pp. 1204-1223*, 2005.

[4] E. A. Lee, "Cyber physical systems: Design challenges," *11th IEEE International Symposium on Object Oriented Real-Time Distributed Computing (ISORC), Orlando*, 2008.

[5] T. Nolte, H. Hansson, and L. L. Bello, "Automotive communications - past, current and future," *10th IEEE Conference on Emerging Technologies and Factory Automation, Vol. 1*, 2005.

[6] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, *et al.*, "Experimental security analysis of a modern automobile," *IEEE Symposium on Security and Privacy (SP), pp. 447-462*, 2010.

[7] C. o. E. V. C. National Research Council (US) and U. Acceleration, *The Safety Promise and Challenge of Automotive Electronics: Insights from Unintended Acceleration*. Transportation Research Board, Vol. 308, 2012.

[8] H. Kopetz, "The time-triggered model of computation," in *Real-Time Systems Symposium, 1998. Proceedings., The 19th IEEE*, pp. 168–177, IEEE, 1998.

[9] W. Steiner, "Candidate security solutions for ttethernet," *IEEE/AIAA 32nd Digital Avionics Systems Conference (DASC)*, 2013.

[10] M. Wolf, A. Weimerskirch, and C. Paar, "Security in automotive bus systems," *Proceedings of the Workshop on Embedded Security in Cars (escar)*, 2004.

[11] Z. Zhang, E. Eyisi, X. Koutsoukos, J. Porter, G. Karsai, and J. Sztipanovits, "A co-simulation framework for design of time-triggered automotive cyber physical systems," *Simulation Modelling Practice and Theory, Vol. 43, pp. 16-33, Elsevier*, 2014.

[12] W. Stallings, *Cryptography and Network Security: Principles and Practices*. 5th Edition, Prentice-Hall Press, 2010.

[13] H. Gilbert and H. Handschuh, "Security analysis of sha-256 and sisters," 2004.