Zero Latency Encryption with FPGAs for Secure Time-Triggered Automotive Networks

Shanker Shreejith*† and Suhaib A. Fahmy*
*School of Computer Engineering
Nanyang Technological University, Singapore
†TUM CREATE, Singapore
Email: {shreejit1,sfahmy}@ntu.edu.sg

***Abstract*—Security has emerged as a key concern in increasingly complex embedded automotive networks. The distributed architecture and broadcast transmission characteristics mean they are vulnerable and provide little resistance to intrusive and non-intrusive attack mechanisms. Incorporating data security using traditional approaches introduces significant latency which can be problematic in the presence of real-time deadlines. We demonstrate how a security layer can be added within the network communication controller in modern time-triggered systems, without introducing additional latency or processing overheads. This allows critical communications to be secured in a manner that is transparent to the processors in the electronic control units (ECUs), while also safeguarding network communication properties.**

I. Introduction and Related Work

Modern vehicles are increasingly automated, with high end vehicles incorporating more than one hundred networked electronic control units (ECUs) that together perform millions of computations per second. As these systems replace mechanical approaches, a major challenge is maintaining reliability of a complex distributed network of ECUs. Researchers have demonstrated the susceptibility of such networks to remote attacks, resulting in an attacker gaining control over vehicle dynamics [1], [2]. While traditional automotive networks were designed to be closed – requiring physical access to employ such attacks – the proliferation of wireless access for entertainment and other core functions provides new pathways for attackers. Furthermore, with technologies like Vehicle to Vehicle (V2V) communication on the horizon, a single infected vehicle might jeopardise the safety of other vehicles on the road.

The security of vehicular systems has been subject to significant research in recent years. Analysis of security requirements in automotive systems was discussed in [3], [4], wherein the authors evaluate threat models based on the architecture of vehicular systems. A survey of threat models (including wireless and non-invasive), risks of traditional networks, and protection mechanisms prevalent in modern cars is presented in [5].

Methods like cryptography, anomaly detection and trusted groups have been proposed to provide data security/authenticity in [5], [6] and others, though these introduce processing overheads and remain vulnerable to replay attacks, where stale captured data is played back over the network. Most proposed approaches integrate security at the application layer, relying on the existing computational structure in the ECU to add security features, thereby not disturbing the network infrastructure. This, however, adds additional load on the ECU, and can result in significant added latencies, which can then impact network characteristics. With a custom network interface, as in the case of an FPGA-based ECU, we can integrate such data security transparently at the network layer, without affecting the real-time guarantees of the time-triggered protocol. The proposed scheme uses enhancements within the network protocol layer, which are made feasible using parallel configurable extensions on the custom network controller, and we suggest this is feasible using reconfigurable hardware as a first step.

FPGAs have been used in driver assistance systems as a way of providing the high performance computation necessary for real time computer vision [7], [8]. More recently, researchers have proposed their use in automotive networks to provide further capabilities and improved reliability [9]. Standards compliant ECUs (AUTOSAR compliant) have also been implemented on FPGAs [10]. Run-time reconfigurability can also enable further system benefits, such as fault-tolerance for critical ECUs [11], [12].

More generally, FPGAs have been widely employed to accelerate cryptographic functions in high performance computing systems where bit level parallelism can be exploited using custom architectures [13], [14]. Within general networking, FPGAs have also been demonstrated as highly capable in packet filtering [15], [16].

In this paper, we show how cryptography can be added to an extensible time-triggered network controller to add a layer of security to vehicular networks, without compromising their real-time characteristics, and in a manner that is seamless to the ECU applications.

II. System Architecture

The key weakness of standard automotive network protocols (and hence controllers) is that no standard mechanism is available to verify the authenticity and timing of a message. However, such features could be integrated into a custom network controller, and verified at the network layer, without intervention of the processing function [17]. Time-triggered networks, that are standard for safety-critical systems already

978-1-4799-6245-7/14/$31.00 ©2014 IEEE

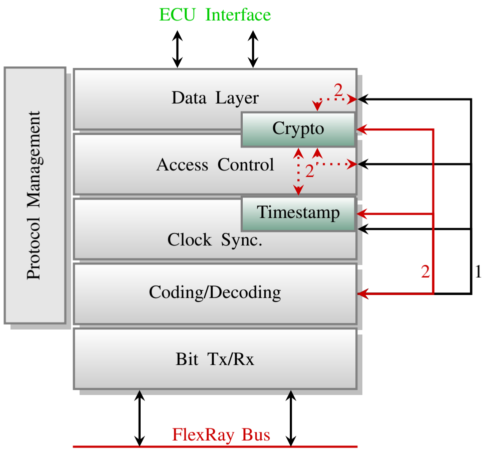

Fig. 1: Time-triggered network enhanced with an intermediate timestamp and data-ciphers.

offer a synchronised view of time at different nodes in the network, and this can be leveraged for securing communication. If messages can be augmented with a timestamp, that identifies when the message was sent, this could allow stale data to be rejected. Adding encryption of messages would allow an ECU to be certain of the source of the message. Our approach enhances the communication controller (CC) by integrating a data cipher into the architecture, as shown in Fig. 1, thus altering the normal datapath (marked **1**), creating the enhanced datapath (marked **2**). Reformatting a data stream to include a timestamp introduces additional entropy to otherwise static data, as proposed in [18] for TCP frames. These extensions are overlapped with standard functions within the CC by extending the datapath using double buffering, prefetching, pipelining, and high wordlength interconnect. Though such extensions could be implemented in software running on the ECU processor, this would result in significant overheads in computation and synchronisation, as well as large variations between the actual time of transmission and the timestamp encoded in the messages.

We present a study using FlexRay, a state-of-the-art time-triggered protocol, to show how the security enhancements can be deterministically integrated into automotive networks. A low-latency PRESENT cipher [19] that can meet the real-time and power/computational constraints imposed by automotive ECUs, is used for the evaluation. Though our experiment uses PRESENT over a FlexRay network, the same principles can be used with other low-latency ciphers and/or time-triggered networks.

A. Secure FlexRay Communication Controller

The FlexRay communication controller (CC) is an implementation of the protocol specification defined by the FlexRay consortium [20]. It allows an ECU to integrate onto the FlexRay bus to exchange messages with other ECUs. The implementation can be a standalone integrated circuit, integrated with micro-controllers as a module, or built using logic in an FPGA.

The FlexRay CC comprises of two functional modules: the Protocol Engine (PE) implements the complete protocol, as defined by the standard, while the Controller Host Interface (CHI) implements a generic interface to the ECU function. The protocol is implemented using the functional sub-blocks, encapsulated within the PE: Clock Synchronisation (CS) that manges synchronisation and timing, Medium Interface Layer (MIL) that handles encapsulation/decapsulation and encoding/decoding of data, and Protocol Management Module (PMM) that controls the PE based on network conditions and ECU commands.

Fig. 2 shows the datapath extensions that implement timestamping and timestamp synchronisation at the interface, integrated into the MIL module of our secure CC. These extensions are possible as we have designed our CC to support extensions in the datapath at the network level [21]. The normal dataflow path in a standard controller design is marked as **NP** in Fig. 2, where the data from the processing logic flows up/down from/to the bitwise decoding/encoding blocks. With our integrated timestamp block, the communication controller establishes a common time-base using techniques defined by the FlexRay protocol during the initialisation phase of the network. A leading coldstart node initialises the start-up procedure where it transmits a start-up frame and waits for other coldstart nodes to join in. Once more than two coldstart nodes integrate, non-coldstart nodes (i.e., the ones which cannot trigger the start-up procedure) adopt this schedule and integrate into the network completing the start-up procedure. We have integrated the timestamp start-up procedure to closely follow the FlexRay clock synchronisation scheme, thus providing a common time-base to all participating nodes, with provisions to correct for drifts.

On the transmit path, the timestamp insertion logic appends timestamp information into the outgoing data, just before it is encoded into the byte-packed format for transmission, all within the CC. Alternatively, if the function is disabled, the ECU processor would instead read the timestamp information and append it to the sensor data before passing it on to the CC for transmission. Similarly, on the receive path, the timestamp processing unit extracts the timestamp information from the decoded data and can be configured to reject incoming data if the timestamp is not current. Alternatively, it can forward the entire packet to the ECU (as in the normal case), where the ECU would compare its timestamp to the current time (from the CC) and decide on its validity.

The timestamped data is encrypted by the cryptographic block using a pre-shared key, then encoded and transmitted bit-by-bit, as per the FlexRay standard. In the receive path, the bits from the physical medium are voted and decoded as per the FlexRay protocol. The cryptographic block deciphers the received data and extracts the timestamp, which is then validated, with the plain text data forwarded upstream. This *cipher enhanced datapath* is shown in Fig. 2 marked as **PP**.

The PRESENT module uses multiple iterations (or rounds) of substitution and permutation operations to convert an incoming data block into cipher text, each round is controlled

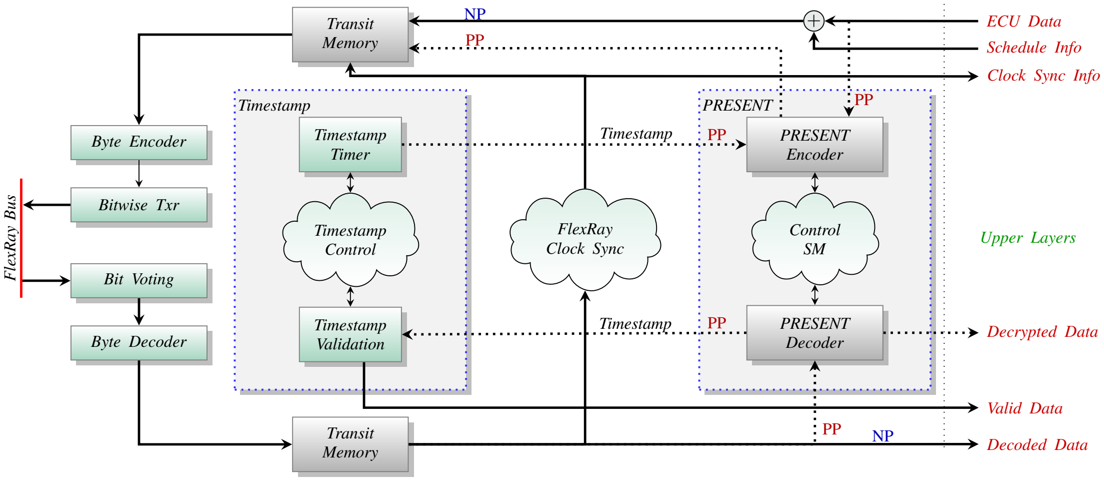

Fig. 2: Datapath extensions: timestamp synchronisation, processing and cipher logic

by a *round key*. The cipher operates on a block of 64 bits and can be configured to have 80 or 128 bit keys and can have up to 65536 rounds per cycle. These options can be changed by the ECU during operation.

The PRESENT module instantiates one buffer each in the transmit and receive direction to store the (prefetched) data from/to the upper layers. 64-bit datapaths are used to ensure that a block is transferred in every single cycle. Once a block of data is available in the buffer, it is read out in one cycle and further encrypted in n cycles, where n is a configurable number of rounds. A key store manages the round keys for each round, and handles the commands to alter cipher properties (no. of rounds, keys) by regenerating the new set of keys in the background using a double buffer. The pipelined operation enables the encryption/decryption to overlap with transmission/reception of previous/subsequent data blocks (or headers), effectively hiding the latency in the buffering latency that is present even with no encryption.

B. Software Encryption

When a powerful computational core (or co-processor) is available in the ECU, data encryption can be carried out within the ECU instead. In such a case, the current time value is read from the CC registers and is used to encode the data by using round-keys generated in the same manner. In the receive path, the decoded timestamp value from the header segment is read into the ECU to decode the data. However, we observe that the timestamps are inaccurate as there is no synchronisation between the communication schedule and tasks schedule on the ECU.

III. Test Setup and Results

To evaluate the proposed scheme, we have integrated four ECUs on a single Xilinx Virtex-6 LX240T FPGA (Xilinx ML-605 development board), connected together using a FlexRay bus channel which is completely contained within the FPGA, replicating an emulated cluster of ECUs [22]. Each ECU in Fig. 3 is served by a local clock and comprises a MicroBlaze softcore processor (each slightly differing in memory configuration) integrated to our secure FlexRay CC. These run a set of tasks that generate incremental data patterns every 100 milliseconds and exchange data in every cycle of duration 10 milliseconds. The enhanced CC consumes up to 86% of resources in each ECU, with the test setup consuming 44% of the Virtex-6 device, and can be clocked up to 124 MHz, more than the 80 MHz required by FlexRay. Table I describes the utilisation of the secure CC in more detail, comparing it against a standard implementation without enhancements. It can be observed that including the PRESENT logic within the controller results in higher resource consumption and marginally higher power consumption than the standard core.

TABLE I: Comparison of CC resources on XC6VLX240T.

Function	Submodule	FFs	LUTs	BRAMs	DSPs
normal CC	MIL × 2	1038	1610	2	0
	Others	3541	5881	12	2
	Total	5617	9068	16	2
secure CC	PRESENT × 2	2548	2274	4	0
	MIL × 2	1275	1463	3	0
	Others	3578	6081	12	2
	Total	11224	13386	26	2
Overhead		5607	4318	10	0
(%)		99.8%	47.62%	62.5%	0%

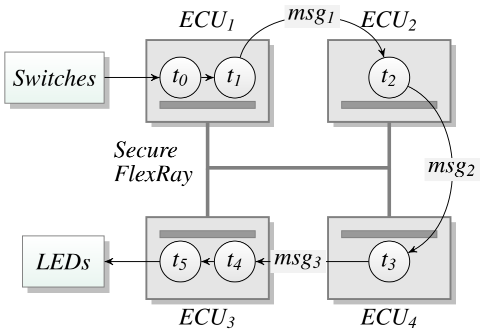

Fig. 3: Test setup with 4 ECUs on ML-605 development board.

TABLE II: Latency of operation per 64 bit data block

Impli. (Rounds, Clock)	Encryption(μs)	Decryption(μs)
HW (up to 470, 80 MHz)	w/in Txn boundary	w/in Rxn boundary
SW MB (32, 100 MHz)	7204	7450
SW ARM (32, 667 MHz)	40.9	42.1

We also evaluate the cost of performing encryption in software as opposed to the integrated approach within the CC. For this, a C implementation of PRESENT is used on the MicroBlaze ECUs. We also evaluated the same on an ARM-based ECU using a Xilinx Zynq platform. The results are shown in Table II. We observe that the integrated approach (in CC) does not incur any additional latency in the transmit or receive direction for handling sensor data encryption, since encryption/decryption are performed during buffering. Meanwhile the software versions require 41 μs and 7 ms on the ARM and MicroBlaze respectively for every 64 bits of data to be encoded or decoded. This delay could be reduced by offloading encryption to a dedicated co-processor like custom IP on the MicroBlaze or the Neon engine on the Zynq. Furthermore, the software implementations rely on additional data exchange (timestamp) between the ECU and CC which results in some inaccuracy, with a worst case slack of a complete network cycle.

IV. Conclusion

Concerns about the security of automotive networks, and demonstrations of serious compromises in modern vehicles are focusing attention on the security of such systems. We have presented an approach to integrating data security in automotive networks that leverages information available at the lower layers of a time-triggered protocol. An ultra lightweight cryptographic function and synchronised timestamps incorporated within the network layer of a custom CC mean we can add security without affecting the real-time nature of such messages. The approach is portable to other time-triggered networks that may replace FlexRay as the backbone for automotive systems. The computational capability and configurable nature of FPGAs enable such configurable levels of security to be integrated into the protocol layers with no loss in determinism, which is impossible to achieve using off-the-shelf components.

We aim to investigate extending the security architecture by incorporating mechanisms to securely generate unique identifiers for each ECU, and exploring partial reconfiguration for altering cipher schemes in response to threats. We are also working on a key-management mechanism that can generate, exchange and revoke keys and policies which can adapt to different threat scenarios, thus providing true dynamic security.

Acknowledgment

This work was supported by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme.

References

[1] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage, "Experimental Security Analysis of a Modern Automobile," in *IEEE Symp. on Security and Privacy (SP)*, May 2010, pp. 447–462.

[2] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno *et al.*, "Comprehensive Experimental Analyses of Automotive Attack Surfaces," in *USENIX Security Symposium*, 2011.

[3] O. Henniger, L. Apvrille, A. Fuchs, Y. Roudier, A. Ruddle, and B. Weyl, "Security requirements for automotive on-board networks," in *Proc. Int. Conf. on Intelligent Transport System Telecommunications (ITST)*, 2009.

[4] D. K. Nilsson, P. H. Phung, and U. E. Larson, "Vehicle ECU classification based on safety-security characteristics," in *Proc. Conf. on Road Transport Information and Control*, 2008.

[5] I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, M. Kaaniche, and Y. Laarouchi, "Survey on security threats and protection mechanisms in embedded automotive networks," in *Proc. Conf. on Dependable Systems and Networks Workshop (DSN-W)*, 2013.

[6] A. Groll and C. Ruland, "Secure and authentic communication on existing in-vehicle networks," in *IEEE Intelligent Vehicles Symp.*, 2009.

[7] N. Alt, C. Claus, and W. Stechele, "Hardware/software architecture of an algorithm for vision-based real-time vehicle detection in dark environments," in *Proc. Design, Automation and Test in Europe (DATE) Conference*, 2008.

[8] C. Claus, R. Ahmed, F. Altenried, and W. Stechele, "Towards rapid dynamic partial reconfiguration in video-based driver assistance systems," in *Proc. Int. Symp. on Applied Reconfigurable Computing (ARC)*, 2010.

[9] S. Shreejith, S. Fahmy, and M. Lukasiewycz, "Reconfigurable Computing in Next-Generation Automotive Networks," *IEEE Embedded Systems Letters*, vol. 5, no. 1, p. 12 to 15, 2013.

[10] F. Fons and M. Fons, "FPGA-based Automotive ECU Design Addresses AUTOSAR and ISO 26262 Standards," *Xcell journal*, vol. Issue 78, pp. 20–31, 2012.

[11] N. Chujo, "Fail-safe ECU System Using Dynamic Reconfiguration of FPGA," *R & D Review of Toyota CRDL*, vol. 37, pp. 54–60, 2002.

[12] S. Shreejith, K. Vipin, S. A. Fahmy, and M. Lukasiewycz, "An Approach for Redundancy in FlexRay Networks Using FPGA Partial Reconfiguration," in *Proc. Design, Automation and Test in Europe (DATE) Conference*, 2013, pp. 721–724.

[13] N. Bourbakis and A. Dollas, "SCAN-based compression-encryption-hiding for video on demand," *IEEE MultiMedia*, vol. 10, no. 3, pp. 79–87, 2003.

[14] J. Fernando, D. Dalessandro, A. Devulapalli, and K. Wohlever, "Accelerated FPGA based encryption," in *The 2005 Cray Users Group Conference*, 2005.

[15] B. L. Hutchings, R. Franklin, and D. Carver, "Assisting network intrusion detection with reconfigurable hardware," in *Proc. IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2002, pp. 111–120.

[16] I. Sourdis and D. Pnevmatikatos, "Fast, large-scale string match for a 10 Gbps FPGA-based network intrusion detection system," in *Proc. Int. Conf. on Field Programmable Logic and Application (FPL)*, 2003, pp. 880–889.

[17] S. Shreejith and S. Fahmy, "Extensible FlexRay Communication Controller for FPGA-Based Automotive Systems," *IEEE Transactions on Vehicular Technology*, vol. To Appear, 2015.

[18] J. Giffin, R. Greenstadt, P. Litwack, and R. Tibbetts, "Covert messaging through TCP timestamps," in *Proc. Int. Workshop on Privacy Enhancing Technologies*, 2003.

[19] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An ultra-lightweight block cipher," in *Proc. Workshop on Cryptographic Hardware and Embedded Systems (CHES)*, 2007.

[20] *FlexRay Communications System, Protocol Specification Version 2.1 Revision A*, FlexRay Consortium Std., December 2005.

[21] S. Shreejith and S. A. Fahmy, "Enhancing communication on automotive networks using data layer extensions," in *Proc. Int. Conf. on Field Programmable Technology (FPT)*, 2013, pp. 470–473.

[22] S. Shreejith, S. A. Fahmy, and M. Lukasiewycz, "Accelerating Validation of Time-Triggered Automotive Systems on FPGAs," in *Proc. Int. Conf. on Field Programmable Technology (FPT)*, 2013, pp. 4–11.