Hardware-accelerated Data Acquisition and Authentication for High-speed Video Streams on Future Heterogeneous Automotive Processing Platforms

(Invited Paper)

Martin Geier
Chair of Real-Time Computer Systems
Technical University of Munich
geier@rcs.ei.tum.de

Fabian Franzen
Chair of IT Security
Technical University of Munich
franzen@sec.in.tum.de

Samarjit Chakraborty
Chair of Real-Time Computer Systems
Technical University of Munich
samarjit@tum.de

ABSTRACT

With the increasing use of Ethernet-based communication backbones in safety-critical real-time domains, both efficient and predictable interfacing and cryptographically secure authentication of high-speed data streams are becoming very important. Although the *increasing data rates* of in-vehicle networks allow the integration of more demanding (e.g., camera-based) applications, processing speeds and, in particular, memory bandwidths are no longer scaling accordingly. The *need for authentication*, on the other hand, stems from the ongoing convergence of traditionally separated functional domains and the extended connectivity both in- (e.g., smartphones) and outside (e.g., telemetry, cloud-based services and vehicle-to-X technologies) current vehicles. The inclusion of cryptographic measures thus requires careful interface design to meet throughput, latency, safety, security and power constraints given by the particular application domain. Over the last decades, this has forced system designers to not only optimize their software stacks accordingly, but also incrementally move interface functionalities from software to hardware.
This paper discusses existing and emerging methods for dealing with high-speed data streams ranging from software-only via mixed-hardware/software approaches to fully hardware-based solutions. In particular, we introduce two approaches to acquire and authenticate GigE Vision Video Streams at full line rate of Gigabit Ethernet on Programmable SoCs suitable for future heterogeneous automotive processing platforms.

1 INTRODUCTION & RELATED WORK

Even though neither Ethernet- nor, more generally, IP-based communication networks were originally intended for safety-critical real-time applications, their flexibility, low component cost and general pervasiveness in information technology have led to widespread use in the industrial automation, avionics and, more recently, automotive domains. Traditionally home to specialized electronic control units (ECUs), the automotive industry has mostly relied on proprietary, relatively low-speed bus systems carrying various higher-level protocols to interconnect the up to 70 ECUs in current vehicles [10]. Maximum bit rates range from 20 kbit/s for LIN (used for non-critical low-cost components) to 500 kbit/s for CAN (safety-critical subsystems such as powertrain, anti-lock brakes and driver assistance) up to 10Mbit/s with FlexRay (handling, e.g., dynamics/stability-related traffic in high-end models) [23].

Although the accumulated data rates of traditional vehicle functions are continuously rising (e.g., from around one hundred kbit/s over two decades ago to several Mbit/s for powertrain and chassis [14, 16]), they are easily surpassed by the throughput requirements of even a single video stream. In the infotainment domain, this motivated the development and integration of MOST, which is capable of handling data rates of 25/50/150 Mbit/s (depending on version and communication medium) and can be used to stream (compressed) movies stored on the head unit to the rear seats or to carry a live feed from a rear-view camera [2, 11]. Advanced Driver-Assistance Systems (ADAS), however, still rely on dedicated point-to-point links based on LVDS (Low-Voltage Differential Signaling) between cameras and corresponding ECUs [11, 12].

The increasing number of bus systems and ECUs facilitating the integration of more advanced features poses various challenges both due to their cost and during development and integration. This, in particular, holds true for the amount of wiring reported to exceed three miles [21], which is becoming prohibitive due to packaging and weight restrictions. For the same reasons, new functionalities may no longer be added following the one-ECU-per-function paradigm and thus might imply careful mapping onto existing architectures. As a result, vendors and academia are moving away from such federated towards integrated architectures [11, 15] commonly relying on centralized ECUs for computation and Ethernet as communication backbone. Although traditional Ethernet based on two twisted pairs of wiring (100BASE-TX) had already been used for diagnostics and updates [8, 12] over the OBD-II (On-Board Diagnostics) port, the introduction of Automotive Ethernet over Unshielded Twisted Single Pair via BroadR-Reach/100BASE-T1 and 1000BASE-T1 now enables the widespread deployment within vehicle backbones [13].

Although specialized Ethernet standards might provide sufficient throughput for future electrical and electronic (E/E) architectures, they – like nearly all traditional automotive bus systems introduced earlier – lack integrated support for security measures such as authentication or encryption. Combined with no, relatively weak or only incompletely implemented

ICCAD '18, November 5–8, 2018, San Diego, CA, USA

ACM ISBN 978-1-4503-5950-4/18/11…$15.00
https://doi.org/10.1145/3240765.3243478

protection features on the individual ECUs, this has led to various attacks on automotive E/E architectures from both within [10] and the outside [3, 21, 22] of the particular vehicles. With such current attacks – capable of triggering false sensor readings, unlocking doors, monitoring of position and cabin up to interfering with engine and brakes – already spanning across previously separated functional domains (such as powertrain, chassis and multimedia with its wireless interfaces), the introduction of vehicle-to-X technologies will most likely expand the attack surface far beyond a single vehicle.

This combination of rising data rates, envisaged ECU centralization and undeniable need for authentication results in a significantly increased demand for processing power on ECUs, which previously have been reported of being unable to handle merely Gigabit Ethernet alone [11, 12]. With similar challenges also present in other embedded system domains, however, selected methodologies might be applicable to current CPU-driven and, in particular, future heterogeneous automotive processing platforms, too. In addition to graphics processing units (GPUs) already used in human machine interfaces [6] and occupancy grid generation [1], the latter are expected to soon integrate hardware-programmable Field Programmable Gate Array (FPGA) structures for, e.g., future situation-adaptive ADAS [4]. This aligns with the general trend in embedded systems to extend a traditional fixed-function System-on-Chip (SoC) with configurable FPGA fabric as the resulting Programmable SoC (pSoC) combines the high efficiency of current SoCs (in terms of high throughput and low power consumption) with the flexibility of FPGAs than can (fully or partially) be reconfigured within seconds.

In this paper, we thus both discuss existing methods for acquiring and authenticating high-speed data streams on traditional SoCs and introduce two hardware-based solutions for current pSoCs. Starting with methods used in most of today's embedded systems and general-purpose computing platforms (such as PCs and smartphones), we investigate how the impact of heavy I/O interface load on the system CPU(s) is mitigated. After a short digression on non-standard image acquisition solutions found in the industrial automation domain, we continue with cryptographic acceleration functionalities available in common CPU cores. In contrast to these traditional methods, we first present a hardware-based solution for high-speed image acquisition from a Gigabit Ethernet (GigE) Vision camera on current heterogeneous processing platforms with configurable FPGA fabric. As demonstrated on a Xilinx Zynq pSoC combining a fixed-function SoC-like Processing System (PS) with FPGA-based Programmable Logic (PL), our solution is capable of handling full GigE line rate without CPU intervention by feeding the extracted video stream to the PL for subsequent authentication, image processing and/or frame-buffering. In addition, we propose a novel I/O architecture for Linux-based GigE Vision gateways aimed at secured redundant networks transporting video streams from several cameras. To this end, we introduce a hybrid star/line network topology and a mixed-hardware/software gateway that generates both authentication and parity data, which are sent to the receiving node over a shared redundancy link.

The rest of this paper is organized as follows. Sec. 2 briefly introduces automotive E/E architectures to summarize challenges and trends on both architectural and node level. We then focus on interface and cryptographic features of CPU-driven nodes (Sec. 3) and on the architecture of heterogeneous processing platforms (Sec. 4.1). The remainder of Sec. 4 then introduces our proposed hardware architectures for acquisition (Sec. 4.2) and authentication (Sec. 4.3) including selected implementation results. Sec. 5 finally concludes our work.

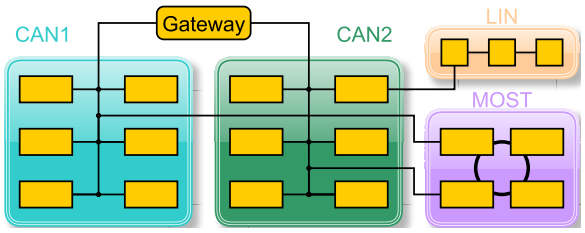


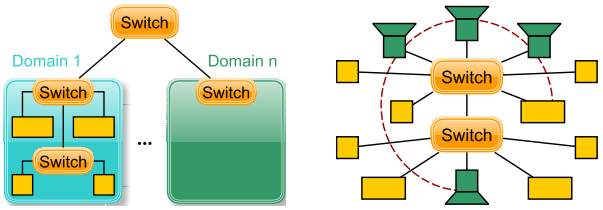
Figure 1: Typical federated E/E architecture [17–19]

2 AUTOMOTIVE E/E ARCHITECTURES

Historically, automotive E/E architectures have been driven by a number of factors aside from security (in terms of both authenticity and confidentiality) [3]. Replacing purely mechanical subsystems with electromechanical solutions not only increased overall efficiency (in terms of reducing both fuel consumption and emissions), but also enabled the integration of various comfort (e.g., central locking and infotainment) and, more importantly, safety features (ranging from anti-lock brakes to complex ADAS) [10, 18]. The industry's distributed development model with Tier 1 suppliers contributing entire functionalities for integration by the vehicle's OEM (Original Equipment Manufacturer) has led to a large number of ECUs and a similarly increasing amount of wiring - both becoming prohibitive due to cost, packaging and weight reasons [15]. Although the proposed centralized implementations might remedy these effects, the resulting tight integration of communication and computation (on fewer bus systems and ECUs, respectively) leads to even higher throughput requirements in contrast to the traditional federated architectures.

2.1 Communication: In-Vehicle Networks

Even far before the genesis of integrated computation architectures, automotive systems relied on shared communication busses to carry signals of more than one function or, later, functional domain. Superseding the dedicated point-to-point wires previously added to replace complex and costly mechanical linkages, shared bus systems soon became a bottleneck in terms of both development complexity (i.e., ensuring an adequately consistent time and data model across all ECUs) and available data rate. Often driven by integration and bandwidth concerns, independent bus systems were initially used to connect the ECUs of one particular function domain [3]. With the advent of more extensive safety and comfort features, those bus systems had to be bridged to facilitate data transfer across domain boundaries. Fig. 1 shows such an architecture with a dedicated gateway between two CAN busses. Even before the introduction of remote telemetry interfaces, additional cross-domain connections were created by ECUs controlling, e.g., multimedia and infotainment functions (such as adaptive volume control or in-tunnel navigation relying on speed and distance measurements from the chassis domain).



(a) Domain-oriented [8, 17] (b) Chassis-oriented [11, 12]

Figure 2: Prospective Ethernet-based architectures with proposed redundancy link (dashed red) in Fig. 2b

Ethernet-based in-vehicle networks, on the other hand, generally have been proposed with homogeneous, fully-switched topologies as reproduced in Fig. 2. Whilst the more traditional domain-oriented topologies recreate the known hierarchical structure (Fig. 2a) with some (sub-)domains potentially still relying on other bus systems, others aim at fully integrated double-star backbones (Fig. 2b) resembling the physical layout of nodes (ECUs, sensors and actuators) within a vehicle.

The increasing number of camera streams fed to both the driver's information screens (e.g., for surround view) and the ECU(s) implementing ADAS functions (e.g., lane departure warning) requires in-vehicle networks to carry dozens (in case of compressed video [12]) up to many hundreds of Mbit/s of video data. Although systems initially relied on uncompressed streams [8], upcoming automotive SoCs for the ADAS domain now include low-latency H.264 codecs to enable transmission of multiple HD streams over a single Ethernet link [20].

2.2 Computation & I/O Interfaces: ECUs

Although ECU architectures not only are inherently proprietary (and thus barely publicly documented), but also heavily depend on the particular functional domain, the underlying microcontrollers or SoCs nonetheless share many similarities with non-automotive embedded systems (Fig. 3). Processor-wise, current architectures range from low-speed single-core CPUs (for, e.g., body applications) via dual-cores with lockstep operation (chassis, powertrain and safety domains) up to asymmetric dual quad-cores with additional lockstep CPUs and video coprocessors for ADAS applications. Non-persistent storage is generally provided by on-chip SRAM (with ECC if required) on both per-core (L1 caches, scratchpad) and shared levels. In addition, some high-end processors not only rely on (less deterministic, but on average faster) multi-level cache hierarchies, but also integrate external interfaces or entire controllers for off-chip storage (such as flash or DDR).

CPUs with their caches, memories and peripherals are then linked by (commonly multiple) on-chip interconnects for data transfers whilst interrupt request (IRQ) lines between peripherals and CPUs enable a timely reaction to various interrupt sources such as timers or I/O peripherals. Direct memory access (DMA) controllers on system and peripheral level enable CPU-independent data transfers to/from memory and are commonly used for (potentially unsynchronized) high-rate streams that otherwise would require immediate and extensive attention of the CPU. Both techniques are widely applied for in-vehicle network interfaces and some I/O peripherals acquiring/generating complex high-rate signals. One particular source of interrupts are the various I/O signals

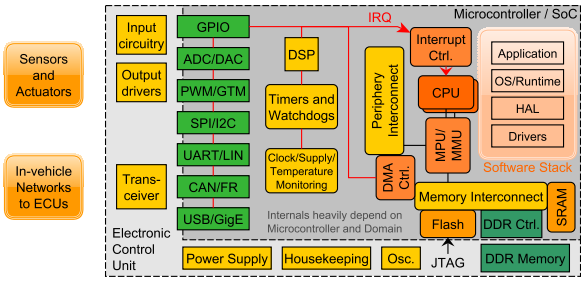


Figure 3: Electronic Control Unit (exemplary)

required for engine control applications relying on precise delays and crankshaft angle measurements to optimally trigger fuel injection and ignition. Until recently, such functionalities were implemented using interrupt service routines (activated by the variable-speed crankshaft), timers and PWM modules and required careful design due to the extremely high interrupt rates (of above 10 kHz). Although multi-core ECUs have been reported to alleviate this situation, many controllers targeting the powertrain domain now include specialized hardware accelerators in the shape of a Generic Timer Module offloading time-critical processing steps from the CPU [7].

2.3 I/O Interfacing: Challenges & Trends

Driven by the fast-paced integration of new and increasingly demanding applications, automotive E/E architectures face significant challenges on both computation and communication levels. On the individual ECUs, processing is no longer handled by (an increasing number of) CPUs alone as several types of hardware accelerators ranging from software-controllable DSPs and GPUs to hardware-programmable FPGA structures and fixed-function offload units (e.g., GTM) are becoming part of current automotive SoCs. In combination with the large external memories supported by certain architectures, they enable complex ADAS applications that perform complex processing steps (such as stereo vision) on multiple video streams. The increasing data rates on incoming interfaces and internal interconnects, however, require careful design to avoid bottlenecks within the device – particularly with regard to the proposed integrated E/E architectures. In addition, both addition of new functionalities and adaption to changing operation and communication environments depend on safe and secure over-the-air updates of ECU firmware and, potentially, reconfiguration of programmable hardware.

Although Ethernet already has been introduced for diagnostics, software updates and, later, video transmission in the infotainment and ADAS domains, an entirely Ethernet-based in-vehicle network will combine various data streams with mixed criticalities onto a single communication medium. On network-level, this requires reliable transmission of critical (e.g., control) messages with guaranteed data rates and limited latencies. In contrast to several proprietary real-time Ethernet variants, the relatively new AVB/TSN (Audio Video Bridging resp. Time-Sensitive Networking) standards extend Ethernet (switches) with time synchronization, stream reservation and enhanced forwarding/queueing capabilities [9]. Many complex Ethernet controllers found on node-level thus support multiple receive and transmit queues to separate

incoming and outgoing data streams according to criticality in terms of latency, throughput and integrity to ensure functional safety. Security features such as authentication at full line rate, however, are not yet commonly implemented. Although several standardized security extensions (such as TLS, IPsec and MACsec) exist for particular layers of the OSI model, it is currently unclear which – if any – solution is suitable for high-speed in-vehicle networks. In addition, the convergence of external (WiFi, vehicle-to-X and cloud) and in-vehicle network towards the Internet Protocol (IP) increases both local and remote attack surfaces and might even necessitate the integration of firewalls or intrusion detection systems known from general-purpose information technology.

3 I/O ON CPU-DRIVEN PLATFORMS

On the majority of today's traditional, CPU-driven processing platforms, both I/O peripherals and hardware accelerators are controlled via (CPU) instructions for regular memory accesses (memory-mapped I/O). Combined with unified address space, this enables CPUs, DMA controllers and other bus masters to transparently transfer both configuration and payload data between system components. The exact mode of operation for data acquisition and authentication then depends on both the particular peripheral and the chosen software environment.

3.1 Hardware/Software Interaction

Data transfers related to I/O peripherals can be driven by either the CPU(s) or a system/peripheral-level DMA controller. Whilst the former results in less complex and more predictable implementations, it also decreases available CPU performance due to time-consuming load/store operations. This holds true for both polled (where the CPU continuously queries the device status) and interrupt-based I/O (where the peripheral actively triggers the CPU for shortest-possible response times). DMA-based data transfers, on the other hand, execute without CPU intervention but require careful synchronization to avoid race conditions. Many network interfaces thus rely on a hybrid approach that dynamically migrates between polling and interrupt-based operation depending on the current CPU load. Network stacks commonly use such Interrupt Moderation in their control path to detect incoming packets, which (if desired) seamlessly integrates with DMA-based data paths.

In case of the NAPI subsystem of the Linux kernel, for instance, the first packet (within a certain period) is received using interrupts. Afterwards, the kernel disables the receive interrupt of the particular network interface and schedules a kernel thread with high priority. As soon as the latter is executed, the interface's receive ring is polled until all packets are processed and the kernel eventually returns to IRQ-based operation. Hardware support for interrupt coalescing can be used in addition, but is not necessary for this approach to work. This mechanism is in stark contrast to the engine control application in Sec. 2.2 where every single interrupt is required.

In addition to those generic acquisition methods, many I/O peripherals contain additional hardware to offload interface- or protocol-specific operations from the CPU(s). In case of Ethernet controllers, for instance, this holds true for the calculation of checksums on multiple layers of the IP protocol for in- and egressing packets (IP/TCP/UDP checksum offload).

Authentication and other cryptographic operations, on the other hand, are available via specialized instruction set extensions of recent CPUs. ARMv8 accelerates not only encryption and decryption in the shape of single AES rounds, but also hashing using SHA1 and SHA256. Recent x86-based CPUs provide equivalent support through AES-NI and SHA-NI. Whilst AES is primarily used for secure message encryption, it can also serve as a primitive for strong cryptographic hash functions. On ARMv8 and x86, those dedicated CPU instructions not only result in almost no runtime overhead, but also are unprivileged and thus directly usable from userspace.

In contrast to extended instruction sets, some platforms (e.g., Marvell ARMADA) feature a dedicated hardware crypto accelerator module accessed using memory mapped I/O.

3.2 Device Drivers & Operating System

Once the hardware has finished data acquisition, control is handed over to the CPU(s) for subsequent application processing – generally supported by either a bare metal runtime environment (RTE) or a fully-fledged Operating System (OS). Whilst both commonly are composed of several layers to handle system complexity (abstraction), only the latter provide both managed access to shared hardware or software services (multiplexing) and strict resource separation for applications.

Though network-level offloading mechanisms such as checksum calculation are mostly transparent and only require driver modifications (for initial configuration), image acquisition and processing pipelines usually span multiple software layers. Once received by the device driver, incoming image data are processed by the network stack and passed to userspace (via, e.g., a socket interface), where they are further parsed by an image acquisition (e.g., GigE Vision) library and forwarded to the actual application, which itself might integrate additional libraries for image processing. As this layering results in significant overhead during acquisition, camera and library vendors commonly include proprietary filter drivers that hook between device driver and network stack to intercept high-rate video data for direct forwarding to the application.

On Linux with locally connected USB cameras, however, both library-based acquisition (via libusb) and a kernel-based alternative are available. In the latter case, image acquisition is fully handled by the kernel's Video4Linux subsystem, which eventually passes entire images to userspace. Although an analogous implementation is conceivable for GigE Vision, commercial Linux acquisition libraries rely on filter drivers.

In case of data authentication, similar implementation options are available that range from userspace-only (e.g., TLS using OpenSSL) via mixed (e.g., OpenVPN using tun/tap devices) to entirely kernel-based data paths in case of IPsec.

4 PROPOSED I/O ARCHITECTURES FOR HETEROGENEOUS FPGA PLATFORMS

With current ECUs already relying on heterogeneous processing for certain tasks, those implementing complex video-based ADAS functions are expected to soon integrate programmable FPGA fabric. We thus use Xilinx' Zynq pSoC as a reference platform for such future programmable nodes (Sec. 4.1) and present I/O architectures for both acquisition (Sec. 4.2) and secured redundant transmission (Sec. 4.3) of video streams.

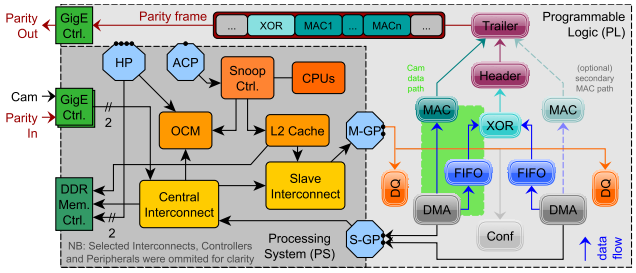


Figure 4: Zynq pSoC with PS (left) and our proposed PL Subsystem for authenticated & redundant Video

4.1 Reference Platform: Zynq pSoC

Current heterogeneous FPGA platforms such as Altera's SoC FPGAs and Xilinx' Zynq pSoCs combine a fixed-function (i.e., hardcore) CPU-driven SoC with configurable FPGA fabric. Whilst the former provides power-efficient software execution, I/O interfacing and accesses to large external memories, the latter enables the system designer to integrate custom softcore blocks to the FPGA fabric. The Xilinx Zynq 7Z020 used for our evaluations features a dual-core Processing System (PS) and an Artix-based Programmable Logic (PL). As shown in Fig. 4, the PS not only contains two ARM Cortex-A9 cores, interconnects and On-Chip Memory (OCM), but also various peripherals for both I/O (e.g., GigE) and storage (e.g., DDR).

Data transfers between PS and PL are enabled by multiple general purpose (GP) and specialized interfaces. Whilst two GP slave (S-GP) ports, four high-performance (HP) and an Accelerator Coherency Port (ACP) are used to access most PS resources from the PL, only the two GP master (M-GP) ports are capable of initiating transactions from PS to PL.

In addition to those already available within the PS, DMA controllers can be instantiated as softcores in the PL. Whilst traditional DMA operation requires persistent (CPU-driven) configuration to handle non-continuous memory buffers, controllers with scatter-gather (SG) capability utilize linear CPU-provisioned buffer descriptor structures to determine required source or destination addresses without CPU intervention.

4.2 Hardware-based Data Acquisition

Although traditional CPU-based processing systems (used, e.g., in industrial automation) are capable of purely software-based image acquisition due to their high-end CPUs, smaller and less power-hungry devices struggle to deal with increasing aggregated data rates of current video streams, which now exceed several Gbit/s. Heterogeneous (hardware-programmable) processing platforms, however, enable novel I/O architectures that handle the acquisition entirely in hardware. The network subsystems of current pSoCs, for instance, can easily be extended and accelerated using SG/DMA Proxying [5], which seamlessly interfaces to the receive data path between integrated standard GigE controllers and system CPU(s).

The proposed approach exploits the SG capabilities of standard GigE controllers to redirect incoming Ethernet frames to a custom core that performs hardware-based separation of video-related and remaining traffic. Whilst the former can either be processed in hardware (e.g., by a subsequent image processing pipeline) or stored in a frame-buffer, the latter is transparently forwarded to the IP stack as if directly received.

To this end, our softcore GigE Vision core replicates a small subset of the (CPU-provisioned) buffer descriptors in the PL, which enables efficient frame filtering for standard GigE controllers without any CPU intervention by directing the controller's receive buffer queue pointer towards our core. Our hardware-based solution results in sub-microsecond acquisition latencies, which are two orders of magnitude shorter than software alternatives on significantly faster platforms.

Further architectural details and comprehensive resource, throughput and latency evaluations can be found in [5].

4.3 Hardware-accelerated Authentication

Just as data acquisition, authentication can also be significantly accelerated by moving selected cryptographic operations to hardware. In contrast to the approaches shown in Sec. 3, our proposed I/O architecture maps the cryptographic operations to neither CPU(s) nor a dedicated HW module under CPU control (which fires an interrupt after processing). Instead, a soft-core subsystem is added to the traditional datapath for network I/O. For the transmit datapath, it generates a Message Authentication Code (MAC) for addition to the video stream. In case of a receive datapath, the subsystem verifies the MAC so that tampered packets can be discarded. The mechanism is comparable to checksum offloading supported by most standard Ethernet controllers.

In addition to authenticating video streams, we propose a shared parity link to increase system availability. As shown in Fig. 2b, we combine star and line topologies such that all cameras are connected to the ADAS ECU (star) and each camera is connected to its respective neighbor. In this setup, the shared parity stream is incrementally created camera-to-camera and eventually reaches the ECU at the end of the line topology (dashed red line in Fig. 2b). This RAID5-like mode of operation allows reconstruction of all video data in case of a single link failure without recovery time. As the video streams are spread over spatially distributed links, this setup is resilient against faults impacting a small area. It should be noted that this setup does not protect against camera faults.

Combining the proposed I/O and network architectures not only extends standard GigE Vision setups with authenticated and redundant transmission, but also exhibits adequately low and predictable latencies for hard real-time operation.

To demonstrate the concept, we developed a gateway node relying on our proposed I/O architecture that interfaces our proposed network topology. It bases on the Xilinx Zynq platform, as presented in Sec. 4.1, equipped with three Ethernet interfaces and running Linux 4.9 as OS with a Linux kernel module providing support for our PL design. Initially, all incoming network traffic is processed normally by the OS and copied over to the system DRAM. Using an existing kernel hook, our module then identifies all packets containing video or parity data and notifies our two DMA cores in the PL about the memory locations of the packets by writing to the Descriptor Queue units (DQ, see Fig. 4). Remaining payload and management traffic is processed by the OS as usual.

If at least one parity packet and one video packet are available, both are simultaneously transferred to the PL. Each DMA core thus directly accesses the DRAM via an S-GP port without any further CPU intervention. In parallel to the

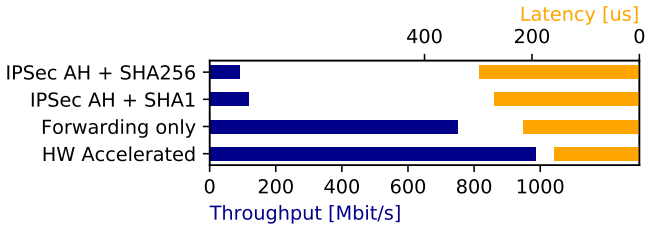


Figure 5: Performance of Authentication Solutions

DMA transfers, the parity and video data streams are XOR-ed together for redundancy. Furthermore, a MAC of the video data is computed and cached until the end of the packet. The Header and Trailer cores embed the updated parity stream in a valid Ethernet frame and append the cached MAC to the existing ones in the packet. As all computations are performed in-stream, the transmission of the new parity packet starts as soon as the updated parity stream is available. The internals of the resulting parity frame are also shown in Fig. 4.

We employ Keccak, the winner of the SHA3 competition, as cryptographic primitive for our MAC. In contrast to the SHA1 and SHA2 families, Keccak's internal design is more suitable for efficient and thus faster hardware implementation.

Our demonstrator is able to process camera data at full line rate of Gigabit Ethernet. Compared to the IPSec tunnel implementation of the Linux kernel, our prototype can handle about ten times more video data as the software-based system with SHA256 MACs (on identical hardware) and, in addition, provides redundancy support. Our solution introduces only 150 us of latency compared to around 250-300 us in case of an IPSec tunnel. Our latency and throughput measurements can be found in Fig. 5. Besides different IPSec MAC algorithms, we also evaluated a simple forward routing scenario that does not modify or authenticate incoming packets and can be used as estimated upper bound of purely software-based solutions.

While our demonstrator implements only the gateway towards the hybrid network topology, our proposed architecture is also suitable for the receiving node where it enables highly efficient processing due to minimized I/O-related interrupts.

5 CONCLUSION

In this paper, we both discussed existing and emerging CPU-driven solutions for acquisition and authentication of high-speed video streams and introduced two highly efficient I/O architectures for current heterogeneous processing platforms. Whilst our hardware-based acquisition solution enables less power-hungry pSoCs to process incoming video streams without any CPU intervention, our authentication gateway adds authentication and redundancy to standard GigE Vision setups via a hybrid star/line network topology. Both approaches not only are capable of full line-rate of Gigabit Ethernet, but also can be combined to increase system resilience against attacks (denial-of-service and spoofing) and single link failures.

REFERENCES

[1] Michael Aeberhard, Sebastian Rauch, Mohammad Bahram, Georg Tanzmeister, Julian Thomas, Yves Pilat, Florian Homm, Werner Huber, and Nico Kaempchen. 2015. Experience, Results and Lessons Learned from Automated Driving on Germany's Highways. *IEEE Intelligent Transportation Systems Magazine* 7, 1 (Spring 2015), 42–57.

[2] Alexander Camek, Christian Buckl, Pedro Sebastiao Correia, and Alois Knoll. 2012. An Automotive Side-View System Based on Ethernet and IP. In *26th International Conference on Advanced Information Networking and Applications Workshops*.

[3] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, Stefan Savage, Karl Koscher, Alexei Czeskis, Franziska Roesner, and Tadayoshi Kohno. 2011. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *20th USENIX Conference on Security*.

[4] Christopher Claus, Rehan Ahmed, Florian Altenried, and Walter Stechele. 2010. Towards Rapid Dynamic Partial Reconfiguration in Video-based Driver Assistance Systems. In *6th International Conference on Reconfigurable Computing: Architectures, Tools and Applications*.

[5] Martin Geier, Florian Pitzl, and Samarjit Chakraborty. 2016. GigE Vision Data Acquisition for Visual Servoing Using SG/DMA Proxying. In *14th ACM/IEEE Symposium on Embedded Systems for Real-Time Multimedia*.

[6] Antje Gieraths, Christoph Müller-Albrecht, and Sean Brown. 2015. Umsetzung der Anforderungen aus der ISO 26262 bei der Entwicklung eines Steuergeräts aus dem Fahrerinformationsbereich. In *Automotive - Safety & Security 2014*. 69–77.

[7] Mukunda Byre Gowda, Carsten Deringer, and Karthikeyan Ramachandran. 2017. Exploring the potential of a multi channel sequencer (MCS) in a next generation GTM-IP using virtual prototypes. In *2nd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology*.

[8] Peter Hank, Steffen Müller, Ovidiu Vermesan, and Jeroen Van Den Keybus. 2013. Automotive Ethernet: In-vehicle Networking and Smart Mobility. In *Conference on Design, Automation and Test in Europe*.

[9] IEEE, Inc. 2010/2011. IEEE 802.1 Time-Sensitive Networking Task Group. http://www.ieee802.org/1/pages/tsn.html.

[10] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. 2010. Experimental Security Analysis of a Modern Automobile. In *IEEE Symposium on Security and Privacy*.

[11] Hyung-Taek Lim, Lars Völker, and Daniel Herrscher. 2011. Challenges in a Future IP/Ethernet-based In-car Network for Real-time Applications. In *48th Design Automation Conference*.

[12] Hyung-Taek Lim, Kay Weckemann, and Daniel Herrscher. 2011. Performance Study of an In-Car Switched Ethernet Network without Prioritization. In *Communication Technologies for Vehicles*.

[13] Kirsten Matheus and Thomas Königseder. 2017. *Automotive Ethernet* (2nd ed.). Cambridge University Press.

[14] Bernd Muller-Rathgeber, Michael Eichhorn, and Hans-Ulrich Michel. 2008. A unified Car-IT Communication-Architecture: Design guidelines and prototypical implementation. In *IEEE Intelligent Vehicles Symposium*.

[15] Marco Di Natale and Alberto Luigi Sangiovanni-Vincentelli. 2010. Moving From Federated to Integrated Architectures in Automotive: The Role of Standards, Methods and Tools. *Proc. IEEE* 98, 4 (April 2010), 603–620.

[16] Nicolas Navet, YeQiong Song, Françoise Simonot-Lion, and Cédric Wilwert. 2005. Trends in Automotive Communication Systems. *Proc. IEEE* 93, 6 (June 2005), 1204–1223.

[17] Arne Neumann, Martin Jan Mytych, Derk Wesemann, Lukasz Wisniewski, and Jürgen Jasperneite. 2017. Approaches for In-vehicle Communication – An Analysis and Outlook. In *Computer Networks*. 395–411.

[18] Thomas Nolte, Hans Hansson, and Lucia Lo Bello. 2005. Automotive communications – past, current and future. In *IEEE Conference on Emerging Technologies and Factory Automation*.

[19] Roman Obermaisser, Christian El Salloum, Bernhard Huber, and Hermann Kopetz. 2009. From a Federated to an Integrated Automotive Architecture. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 28, 7 (July 2009), 956–965.

[20] Renesas Electronics Corporation. 2015. Product Specifications of the R-Car T2. https://www.renesas.com/eu/en/solutions/automotive/soc/r-car-t2.html#specification. [Rev. 09.09.2015].

[21] Ishtiaq Rouf, Rob Miller, Hossen Mustafa, Travis Taylor, Sangho Oh, Wenyuan Xu, Marco Gruteser, Wade Trappe, and Ivan Seskar. 2010. Security and Privacy Vulnerabilities of In-car Wireless Networks: A Tire Pressure Monitoring System Case Study. In *19th USENIX Conference on Security*.

[22] Dieter Spaar. 2015. Beemer, Open Thyself! – Security vulnerabilities in BMW's ConnectedDrive. *c't Magazin für Computertechnik* 05 (2015). https://heise.de/-2540957 pp. 86–89, [English version].

[23] Helge Zinner, Josef Noebauer, Thomas Gallner, Jochen Seitz, and Thomas Waas. 2011. Application and Realization of Gateways Between Conventional Automotive and IP/Ethernet-based Networks. In *48th Design Automation Conference*.