IEICE Communications Express, Vol.13, No.8, 347–350

ComEX

LETTER

Implementation and evaluation of lightweight in-vehicle security switch with hierarchical hash table

Yuji Yano[1, a)], **Hisashi Iwamoto**[2], **Tsutomu Sasao**[3], **and Shingo Ata**[4]

Abstract As a result of the growth of autonomous driving technology, vehicle networks increasingly require not only broadband capabilities and real-time communication, but also enhanced security features. In this paper, we propose a lightweight security switch using hierarchical hash table to achieve security functionality at the central gateway within the vehicle network. Our proposed switch combines header information from Layer-2 to Layer-4 to determine whether incoming packets are attack packets or not. It also achieves fast packet forwarding using low-latency hierarchical hash table. We evaluated the system performance with the FPGA board implementing the proposed security switch, and achieved low-latency transfers of 500 nanoseconds or less.

Keywords: Intrusion detection system, security, hash table, FPGA
Classification: Network system

1. Introduction

In recent years, connected cars have been attracting remarkable attention that connect to IP network. Connected cars will not only be able to receive various network services inside the car, but also be possible to collect dynamically changing vehicle driving information on the cloud and provide feedback for predicting traffic jams and autonomous driving. On the other hand, cyberattacks against vehicles have appeared and improving vehicle security has become an urgent issue. In addition to traditional intrusion attacks on vehicle networks via vehicle debug ports, it is predicted that cyberattacks against vehicle networks via public wireless and Wi-Fi networks will increase.

In conventional IP networks, countermeasures against cyberattacks have been taken using firewalls, intrusion detection systems, virus software, etc. However, as the processing performance of PCs or servers improves, cyberattacks have been larger and more sophisticated, and security devices are currently requiring more CPU and memory resources to protect their own system. Furthermore, when considering the cyber security of vehicle networks, it is desirable that security devices are battery-powered, low-power consumption, small and lightweight. Therefore, we investigated the feasibility of applying lightweight, low-power consumption security technologies used in conventional IP network into in-vehicle networks.

2. Lightweight in-vehicle intrusion detection system

In vehicles, basic modules such as brakes and accelerators have traditionally been interconnected with a vehicle network protocol called CAN (Control Area Network). CAN is an old communication standard designed specifically for controlling vehicles. Therefore, it features low latency and narrow bandwidth but is unsuitable for large-capacity data communication.

In addition, with the recent development of navigation systems and advanced automated driving systems (ADAS), new communication standards such as LIN/FlexRay/CAN-FD/EthernetAVB (EthernetTSN) have appeared as in-vehicle networks. With the advent of new communication standards, the configurations of in-vehicle networks are required to change from the past. Two typical examples are shown below.

(1) Integrating all vehicle networks, including traditional CAN, with Ethernet-based networks.
(2) Operating a hybrid of CAN and Ethernet-based networks [1].

In the second method, a central gateway (CGW) has been proposed and put into practical use to interconnect multiple networks within a vehicle efficiently and reliably.

If we need to implement advanced security functions into the central gateway, short packet performance is important because a large number of sensor devices are connected to an in-vehicle network, or a huge amount of sensor information frequently flows over the network. Furthermore, since vehicles are mobile entities, it's crucial to minimize latency.

Figure 1 shows an example of a vehicle network system that interconnects an Ethernet or IP network and CAN with a central gateway. Here, the IP network domain includes advanced autonomous driving systems (ADAS), navigation systems, communication systems, and in-vehicle cameras. And the CAN domain includes engines, brakes, powertrains, body sensors, etc. All packets in the IP network domain are transmitted and received through the security switch in the central gateway, and the security switch inspects the packets to discover attack packets or abnormal packets.

In this paper, we implement the packet filtering engine into a security switch. The engine has a whitelist table that com-

[1] Graduate School of Engineering, Osaka City University, Sumiyoshi-ku, Osaka-shi, Osaka 558–8585, Japan
[2] Development Dept., poco-apoco Networks Co., Ltd., Chuo-ku, Kobe-shi, Hyogo 650–0023, Japan
[3] Dept. of Computer Science, Meiji University, Tama-ku, Kawasaki-shi, Kanagawa 214–8571, Japan
[4] Graduate School of Informatics, Osaka Metropolitan University, Sumiyoshi-ku, Osaka-shi, Osaka 558–8585, Japan
[a)] d15tb551@st.osaka-cu.ac.jp

DOI: 10.23919/comex.2024XBL0093
Received May 8, 2024
Accepted May 29, 2024
Publicized June 11, 2024
Copyedited August 1, 2024

This work is licensed under a Creative Commons Attribution Non Commercial, No Derivatives 4.0 License.
Copyright © 2024 The Institute of Electronics, Information and Communication Engineers

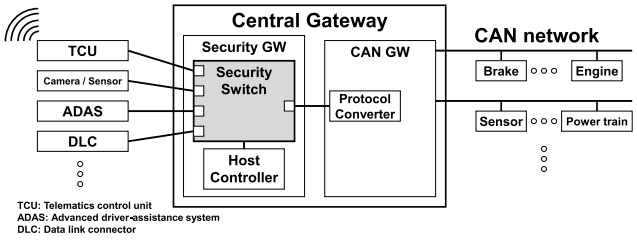


Fig. 1 Example of central gateway.

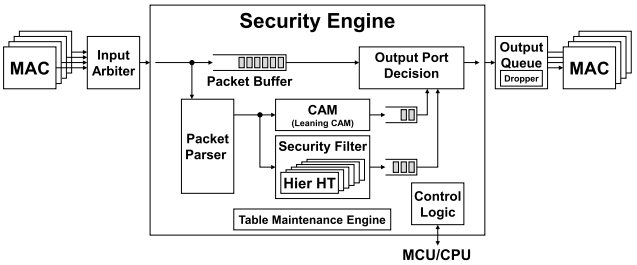


Fig. 2 Proposed lightweight security switch.

bines header information from L2 to L4 as a countermeasure against attacks on vehicles in the IP network domain.

2.1 In-vehicle security switch

Figure 2 shows an overview of the proposed lightweight in-vehicle security switch system equipped with a hierarchical hash table. A security switch consists of an input arbiter circuit that arbitrates input packets, a packet buffer, a packet parser, CAM (Content Addressable Memory), a security filtering engine, an output port determination section, and an output buffer. After the header information is extracted in the packet parser, a FIB (Forwarding information base) table lookup is performed in the CAM, and a whitelist table search is performed in the security filtering engine simultaneously. The security filtering engine checks the header information from L2 to L4 in the packet against a preset whitelist table and decides whether to forward or discard the packet. By implementing this whitelist table in a hierarchical hash table, we can achieve low power consumption and low latency table searches. Additionally, this security switch is equipped with a digital serial interface for table maintenance and debugging/monitoring.

2.2 Hierarchical hash table

In this section, we consider the applicability of CAM, tree search, and hash search as search engines for the security filters. CAM is an SRAM-based search memory, and features high-speed search by fully parallel comparison processing in the memory array. However, while CAM can achieve high search performance, it consumes high power during search operation [2].

Tree search circuit consists of a memory such as SRAM/DRAM, and a comparator. It achieves table search by repeating data reading and pattern match processing. Tree search achieves a smaller footprint and lower power consumption compared to CAM, but it requires many memory reads for one search, resulting in a low search performance and complicated table maintenance. Hash search requires SRAM or DRAM, a hash creation circuit, and a comparison circuit.

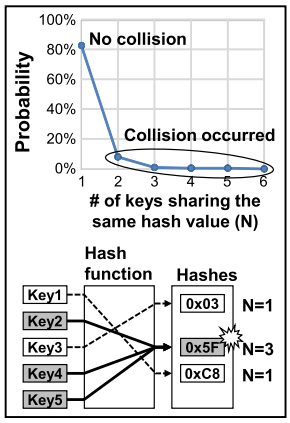


Fig. 3 Hash collision occurrence.

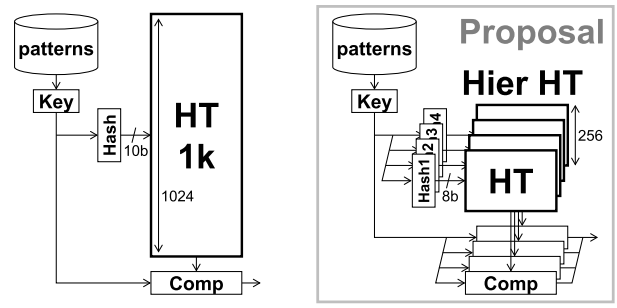
Table I Search engine comparison

Engine	CAM	Tree-based Search	Hash-based Search	
			Hash Table	Count. Bloom Filter
Area	Large	Middle	Large	Small
Latency	Low	High	Low	Low
Power	High	Low	Low	Low
Maintenance	Easy	Hard	Middle	Middle
False positive	-	-	-	Exist

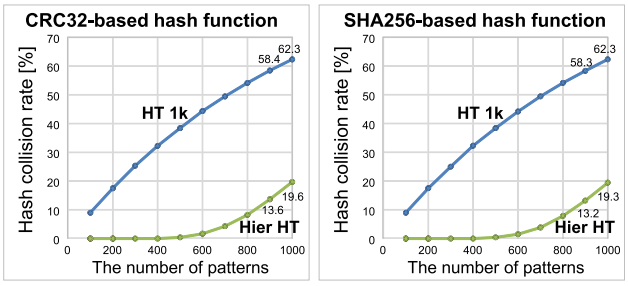
Because the search can be executed with one memory read, it can perform high-speed searches, and features easier table maintenance than the tree search. However, hash searches have a hash collision problem. Since there is a trade-off relationship between the hash collision rate and memory capacity, the hash collision rate can be reduced by using a memory capacity larger than the amount of data actually stored. Therefore, hash search requires a larger memory capacity than tree search, but it is considered the most desirable for in-vehicle security applications because it allows for a search engine with low latency and low power dissipation.

The hierarchical hash table that we propose is a type of hash table that decomposes a lookup table into multiple sub-tables and performs a hash calculation with a different hash function for each table [3]. In our examination of hash collisions, we observed that the vast majority of keys with identical hash values are limited to two or fewer. Furthermore, the probability of three or more keys sharing the same hash value is notably low, as shown in Fig. 3. So the hash collision rate can be reduced by decomposing one large hash table into multiple smaller hash tables. Counting Bloom Filter (CBF) [4] is the most lightweight implementation method to compress memory capacity by storing only count values in memory instead of data when registering data patterns in a hash table. However, CBF has a false positive feature: there is a possibility that a normal packet is treated as an invalid packet and discarded although the probability is low (Table I).

An example of the structure of a hierarchical hash table is shown on the right side of Fig. 4(a). In this figure, four hash tables are connected in parallel, and each table is assigned a unique hash function. When registering a rule in memory, the system calculates four memory addresses in advance, then the rule is written in a vacant entry on a memory block without hash collisions. During a search, four tables are read in parallel, and the four read data are compared with

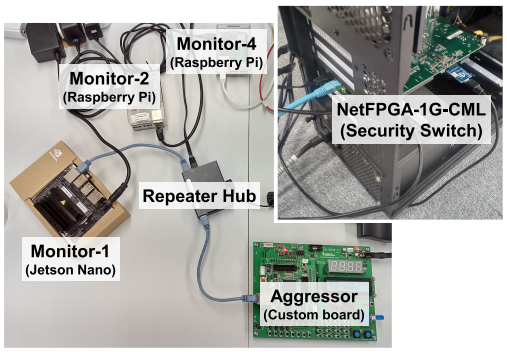


(a) Hash table implementation

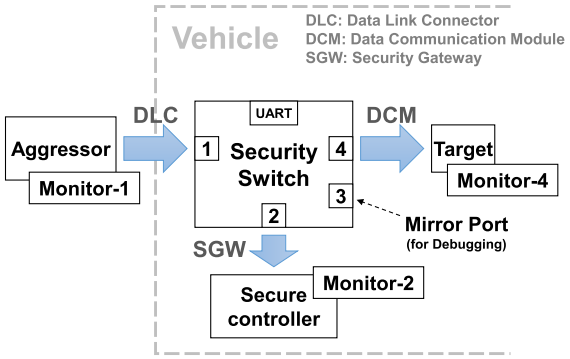


(b) Simulated hash collision rate

Fig. 4 Hierarchical hash table as a search engine.



(a) Fuzzing test environment



(b) Attacks from out-vehicle network

Fig. 5 Fuzzing test environment.

the search key to determine the final match/mismatch.

The hash collision rate in the system of Fig. 4(a) with CRC32 and SHA-256 hash functions was investigated. The results are shown in Fig. 4(b). Both CRC32 and SHA-256 are constructed using repeated shift operations and bitwise exclusive OR (XOR) calculation, so that they are easy to implement in hardware. The outputs of the original CRC32 and SHA-256 hash functions are 32 and 256 bits wide, respectively, so that additional XOR operations are required to reduce them to 10 or 8 bits to match the hash table size. In Fig. 4(b), regardless of the number of registration patterns in the hash table, the hierarchical hash table achieves lower collision rates. Eventually, we implemented a custom bitwise XOR-based lightweight hash function into the system in Fig. 2, because the packet header information has a limited length of a few hundred bits. Furthermore, since in-vehicle networks require few rule updates, a collision-free hash table update may be achieved by dynamically changing the hash functions.

3. Implementation and performance evaluation

We implemented the proposed security switch on the commercial FPGA [5] and evaluated its packet forwarding performance when attacking packets were applied. Figures 5(a) and (b) are a photograph of the evaluation environment for testing the proposed switch, and a diagram of the assumed vehicle environment, respectively. Port-1 of the security switch is the external DLC (Data Link Connector), and port-4 is the internal DCM (Data Communication Module), and a computer board for packet observation is connected to each port. In this evaluation environment, by changing the connection method between the attack packet generator and the security switch, it is possible to perform system evaluations assuming attack tests not only from outside the vehicle but also from inside the vehicle.

We generated a whitelist table within the security switch and performed an evaluation test on the filtering function. Here, packets designed for evaluating the filter function on the switch were sent from the attack packet generator. There were two types of test packet patterns: normal frames and abnormal frames. The fuzzing test packet patterns were transmitted 100 times from the generator, and the transmission cycle was 500 microseconds. All test packets passed the validation test, confirming the normal operation of the whitelist table.

Next, Table II shows the evaluation results of the processing latency of the security switch. We measured the average delay time of four types of communication routes: from the outside DLC to the inside DCM, from the outside DLC to the SGW, from the inside DCM to the outside DLC, and from the inside DCM to the SGW. In Table II, the measured average delay time is the processing latency calculated from the packet capture log acquired inside the security switch, and represents the processing delay time from the input arbiter circuit to the output buffer section in Fig. 2. Then, we estimated the average MAC to MAC delay time as the overall delay time of the security switch. First, we estimated the number of processing cycles from the MAC to the input arbiter circuit and the number of processing cycles from the output buffer section to the MAC when there is no packet congestion through circuit simulation, and added them to the measurement results to determine the overall delay time. As a result, we found that it was possible to provide a security filter function with a processing delay time of approximately 500 nanoseconds or less for various types of attack packets.

Table II Measured/Estimated latency in fuzzing test

Frame Type	Frame Size	Frame Count	Transmission Interval	Measured Latency (Core only)	Estimated Latency (MAC to MAC)
Syn flood	66 B	100	500 µs	384.2 ns	484.2 ns
DHCP snooping	342 B	100	500 µs	386.6 ns	486.6 ns
Sequential ARP	60 B	100	500 µs	383.6 ns	483.6 ns
DIA attack	42 B	100	500 µs	383.8 ns	483.8 ns
ICMP attack	70 B	100	500 µs	386.8 ns	486.8 ns

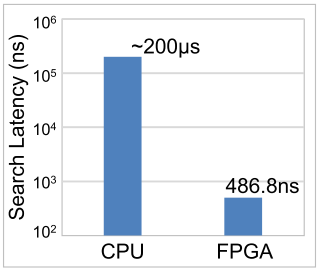


Fig. 6 Search latency.

It indicates that the proposed security switch achieves a speedup of more than 400 times compared to software processing on a CPU (Fig. 6).

4. Conclusion

We proposed a lightweight security switch with a hierarchical hash table to implement security functions at the central gateway in future vehicle networks. The security switch combines header information from L2 to L4 to determine whether it is an attack packet or not, and is capable of forwarding the packet with low latency. We implemented a security switch on the FPGA board and evaluated its performance, confirming low latency transfer of less than 500 nanoseconds.

References

[1] P. Hank, S. Müller, O. Vermesan, and J. Van Den Keybus, "Automotive Ethernet: In-vehicle networking and smart mobility," 2013 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1735–1739, 2013.

[2] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: A tutorial and survey," *IEEE J. Solid-State Circuits*, vol. 41, no. 3, pp. 712–727, 2006. DOI: 10.1109/jssc.2005.864128

[3] T. Sasao, *Index Generation Functions*, Springer Nature, pp.1–165, 2019.

[4] L. Fan, P. Cao, J. Almeida, and A.Z. Broder, "Summary cache: A scalable wide-area web cache sharing protocol," *IEEE/ACM Trans. Netw.*, vol. 8, no. 3, pp. 281–293, 2000. DOI: 10.1109/90.851975

[5] "Home NetFPGA 1G CML," https://github.com/NetFPGA/NetFPGA-public/wiki/Home-NetFPGA-1G-CML, 2020.