CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Information Technology

Proceedings of the 12th Prague Embedded Systems Workshop

June 27 - 29, 2024

Horoměřice

Czech Republic

ISBN 978-80-01-07303-2

Editors:

prof. Ing. Hana Kubátová, CSc.

doc. Ing. Petr Fišer, Ph.D.

Ing. Jaroslav Borecký, Ph.D.

Message from the Program Chairs

The Prague Embedded Systems Workshop is a research meeting intended to present and discuss students' results and progress in all aspects of embedded systems design, testing, and applications. It is organized by members of the Department of Digital Design at the Faculty of Information Technology and supported by the Czech Technical University in Prague. The workshop is focused mainly on new technologies and methods, dependable and low-power design, embedded security, network monitoring and measurements, and algorithms and methods for anomaly detection. The workshop aims to enhance collaboration between universities, not just within the EU. It will be based on oral presentations, mutual communication, and discussions.

There are three types of students' submissions and presentations:

- Full papers describing the students' original research. These papers were submitted to a standard reviewing process.
- Abstracts of authors' earlier published and successfully presented papers (at conferences, in journals, etc.). These contributions were not reviewed; emphasis was put on the presentation and discussion.
- Student posters - abstracts of defended Bc. and MSc. theses with subsequent poster presentation. This workshop session is traditionally organized as a contest sponsored by IEEE and industry.

Ten papers were submitted for the oral presentation, from which five full papers and four abstracts were accepted. This year, contributions from Austria, Germany, and the Czech Republic will be presented.

The technical program is also highlighted by four keynote speakers:

- Colorful like a Chameleon: (In)Security of Wireless Access Control Systems.
  Speaker: Timo Kasper (Ruhr-University, Bochum, Germany; Kasper&Oswald GmbH.) and Tomáš Přeučil (FIT, CTU in Prague)
- Asynchronous Circuits – Old Iron or Enabler for a New Resilience Level of Digital Circuits?
  Speaker: Andreas Steininger (Vienna University of Technology, Austria)
- Digital simulator: from the RTL to the full chip simulations of a low power SoC ASIC.
  Speaker: Jakub Šťastný (ASICentrum, s.r.o.)
- Security Issues in Cyber Physical Cognitive Systems.
  Speaker: Virendra Singh (Indian Institute of Technology Bombay, Mumbai, India)

PESW 2024 program contains four Industrial talks and the Poster sessions with the IEEE contest.

We would like to thank to our sponsors Racyics, ASICentrum, UJP PRAHA, STMicroelectronics, SYSGO, IMA, daiteq, Tropic Square, METIO Software.
Special thanks go to IEEE: IEEE Student Branch at Czech Technical University in Prague and IEEE Young Professionals, organizing student contest, and Czechoslovakia Section of IEEE.

We wish the 12th Prague Embedded Systems Workshop many heated discussions and possible establishment of mutual research cooperation.

Hana Kubátová and Petr Fišer
27th June 2024

Committees

Workshop Chairs

Hana Kubátová, CTU in Prague (CZ)

Petr Fišer, CTU in Prague (CZ)

Programme Committee

A. Bernasconi, Università di Pisa (IT)

J. Bělohoubek, CTU in Prague, FIT (CZ)

A. Bosio, École Centrale de Lyon (FR)

L. Cassano, Politecnico di Milano (IT)

G. DiNatale, TIMA, Grenoble (FR)

J.L. Gaudiot, University of California, Irvine (USA)

V. Janíček, CTU in Prague, FEE (CZ)

K. Jelemenská, STU Bratislava (SK)

M. Jenihhin, Tallinn Univ. of Technology (EE)

P. Kitsos, TEI West. Greece (GR)

M. Krstić, IHP, Frankfurt (Oder) (DE)

R. Kvaček, ASICentrum, Prague (CZ)

F. Leporati, Univ. di Pavia (GR)

I. Levin, Tel-Aviv University (IL)

A. McEwan, University of Leicester (UK)

P. Mróz, University of Zielona Gora (PL)

M. Novotný, CTU in Prague, FIT (CZ)

A. Orailoglu, UC San Diego (USA)

Z. Plíva, TU Liberec (CZ)

M. Poupa, University of West Bohemia, FEE (CZ)

P. Puschner, Vienna University of Technology (AT)

J. Raik, Tallinn Univ. of Technology (EE)

J. Schmidt, CTU in Prague, FIT (CZ)

M. Skrbek, CTU in Prague, FIT (CZ)

J. Sobotka, CTU in Prague, FEE (CZ)

B. Steinbach, TU Chemnitz (DE)

A. Steininger, Vienna University of Technology (AT)

J. Strnadel, Brno University of Technology, FIT (CZ)

Z. Vašíček, Brno University of Technology, FIT (CZ)

W. Zając, Jacob of Paradies University (PL)

Special Session on Network Security Chair

Tomáš Čejka, CTU in Prague (CZ)

Student Poster Session Co-Chairs

Josef Koumar, CTU in Prague (CZ)

Jaroslav Pešek, CTU in Prague (CZ)

Organizing Committee

H. Kubátová, CTU in Prague (CZ)

P. Fišer, CTU in Prague (CZ)

J. Borecký, CTU in Prague (CZ)

M. Novotný, CTU in Prague (CZ)

Secretariat

R. Kinc, AMCA (CZ)

B. Dufková, AMCA (CZ)

Contents

Keynotes

Colorful like a Chameleon: (In)Security of Wireless Access Control Systems

Speakers: **Timo Kasper** *(Ruhr-University, Bochum, Germany; Kasper&Oswald GmbH.)* and **Tomáš Přeučil** *(FIT, CTU in Prague)*

Wireless embedded devices have become omnipresent in applications such as access control (to doors or to PCs), identification, and payments. The talk reviews the security of several commercial devices that typically employ cryptographic mechanisms as a protection against ill-intended usage or to prevent unauthorized access. A combination of side-channel attacks, reverse-engineering and mathematical cryptanalysis helps to reveal and exploit weaknesses in the systems that for example allow opening secured doors in seconds. At hand of real-world examples and live demos, the implications of a key extraction for the security of the respective contactless application are illustrated. As a powerful tool for security-analyzing and pentesting NFC and RFID systems, the open-source project “ChameleonMini” is presented: Besides virtualization and emulation of contactless cards, the device allows to log the NFC communication, and in its latest Revision G acts as an active RFID reader to copy contactless cards on-the-fly.

Timo Kasper

Dr.-Ing. Timo Kasper is executive director of Kasper&Oswald GmbH (KAOS), founded in 2012 together with Prof. Dr.-Ing. David Oswald, offering innovative products and various services for (embedded) security engineering. Timo has studied electrical engineering and information technology at the Ruhr-University Bochum, Germany and at the University of Sheffield, UK. His Diploma thesis (2006) and his PhD thesis (2012) were awarded with a first prize in IT Security. Timo’s field of expertise covers the security of embedded cryptographic systems, such as smartcards, RFID and other (wireless) technology, including penetration testing and implementation attacks.

Tomáš Přeučil

Tomáš Přeučil, MSc. is a PhD student at the Faculty of Information Technology (FIT) at the Czech Technical University in Prague. His research focuses on the security of pervasive devices including access control systems and RFID card security, as well as attacks on non-IP-based networks.

Asynchronous Circuits – Old Iron or Enabler for a New Resilience Level of Digital Circuits?

Speaker: **Andreas Steininger** *(Vienna University of Technology, Austria)*

While the synchrony obtained by a global clock simplifies design and implementation of digital circuits considerably, it also constitutes a strong assumption. This becomes perceivable by the clock distribution problems in high-speed circuits, but also by the uncontrolled error behavior that synchronous circuits usually exhibit upon even a small timing violation. Due to their much more flexible and self-regulated timing, asynchronous circuits do not need a sophisticated clock tree and can accommodate timing variations and delay-related faults naturally. In this talk we will survey approaches to complement this important time-domain property though explicit value-domain fault-tolerance provisions on coding- and circuit-level. In addition, we will investigate how the fail-stop property inherent to asynchronous circuits can be leveraged for easy recovery after repair of a permanent fault, and hence forms an interesting foundation for building self-repairing circuits.

Andreas Steininger

Andreas Steininger studied Electrical Engineering at TU Wien where he also finished his PhD thesis in 1994, and is now working as an Associate Professor at the Department of Computer Engineering. He has been involved in many industrial and scientific projects concerned with real-time communication networks, the design of fault-tolerant / radiation-tolerant computer architectures and their evaluation by means of fault-injection, and testing. His current research focuses on asynchronous (“clockless”) logic design, timing-domain interfacing, metastability, and GALS architectures. He has published over 190 papers in journals and at international conferences and is co-inventor of over 10 patents. He has supervised more than 20 successful PhD theses and serves as the Director for the Vienna PhD School of Informatics and as chair of the Doctoral College Resilient Embedded Systems. Three of his master students have been winners of the faculty’s highly competitive "Distinguished Young Alumnus Award" for the best diploma thesis.

Digital simulator: from the RTL to the full chip simulations of a low power SoC ASIC

Speaker: **Jakub Šťastný** *(ASICentrum, s.r.o.)*

Digital simulator is the basic tool to examine the behavior of the designed digital block. However, digital simulator can support designer's work also beyond the purely digital world - the whole ASIC can be modeled in it on the system level including analog blocks and power supply networks. During the talk we will discuss challenges brought by the full low-power SoC ASIC simulation and present a handful of case studies how we utilized the capabilities of a modern digital simulator.

Jakub Šťastný

Jakub Šťastný studied at the Czech Technical Unverisity in Prague, Faculty of Electrotechnical engineering. He has been working for ASICentrum spol. s r.o. (EM Microelectronic) since 2002, currently at the position of the ASICentrum's Motion and Optical Sensing department leader. During his career he has been working on tens of custom ultra low-power ASIC projects mainly as project manager and digital designer, dealing with devices ranging in size from simple senzor chips to SoC systems.

Security Issues in Cyber Physical Cognitive Systems

Speaker: **Virendra Singh** *(Indian Institute of Technology Bombay, Mumbai, India)*

Migration to Society 5.0 has mandated to go for Industry 5.0 whose priority is to utilize human and machines synergistically. In order to achieve the above stated objective, Cyber Physical Systems (CPS) has become the central part of the Industry 5.0. Industry 5.0 demands cognitive computing along with the Cyber Physical Systems. Industry 5.0 aims at utilizing human and machine synergistically.
On the other hand, with sophisticated cyber attacks all over the world, it is clear that the attackers are well-funded through organized crimes, nation–support, etc. Thus, it has become important to address cyber threat intelligence to prevent some of the ulterior motives of the attackers to use attacks as weapons, in particular, when most of the public infrastructures are driven by sophisticated IT systems and further with the policy of building several smart-cities to address various societal issues. The latter naturally will lead to growth of Internet of Things (IoT), which in turn will increase the attack surface of the underlying infrastructure due to their vulnerabilities, malware susceptibility, and an emergence of denial-of-service (DoS) attacks will be acutely felt. Therefore the system must aims at the infrastructures with proactive sensing of cyber-physical systems using data from physical sensors and integrated from other relevant resources, combined via a broad based intrusion alert system and architecture, adaptable/tunable for a spectrum of applications like, attack predictions in the context of vulnerabilities, security alerts in IoT/SCADA, insider attack correlations etc. To realize properties of speed and accuracy using intelligence from a spectrum of resources, use cognitive security solutions using AI/Deep Learning Systems. This project also envisages the development of techniques of privacy-preserving merging/integrating different datasets and privacy preservation training.

Virendra Singh

Virendra Singh obtained Ph.D in Computer Science from Nara Institute of Science and Technology (NAIST), Nara, Japan. Currently, he is serving as a faculty member at Indian Institute of Technology (IIT) Bombay jointly with the Dept. of Electrical Engineering, and the Dept. of Computer Science and Engineering. Prior to join IIT Bombay, he served as a faculty member at Supercomputer Education and Research Centre (current Computational and Data Science department), Indian Institute of Science (IISc), Bangalore from 2007 to 2011. He also served Central Electronics Engineering Research Institute (CEERI), Pilani, as a Scientist from 1997 to 2007. His research interests are VLSI Design, verification and Test, High Performance Computer Architecture, Cyber security, cyber physical cognitive systems, trustworthy AI, formal verification. He has published about 195 research papers in various journals and international conferences. He is a convener of India-Japan joint research hub on Trustable cyber physical cognitive systems. He is leading a major project on development of AI powered adaptive cyber defence systems funded by government of India.

Industrial Talks

Designing market ready energy efficient silicon in the first shot

Speaker: **Marcus Pietzsch** *(Racyics GmbH., Germany)*

Time to market is key for the success of start-up technologies and groundbreaking new ideas from academic.
Being able to unwrap and showcase the full potential and performance of innovative solutions in integrated silicon introduces the need to deal with the multidimensional, complex challenges of SoC design. This talk presents a disruptive approach to overcome these struggles.
"makeChip" is enabling inventors to keep focus on their core IP while being able to demonstrate PPA (Power, Performance, Area) optimization at product level in the first shot. Technical approaches for designing first time right ultra low power SoC will be presented and discussed along example success stories.

Marcus Pietzsch

Marcus Pietzsch studied electrical engineering at the Technical University Dresden. He was working in academic and industrial IP and chip design within different domains such as communication, medical or automotive. In 2022, he joined Racyics GmbH as "Head of SoC Design".

The Czech Republic and Its Active Contribution to International Semiconductor Strategy Activities

Speaker: **Milan Semmler** *(UJP Praha a.s., Czech Republic)*

The Czech Republic, as one of the developed European economies, is striving to rapidly engage in the dynamically evolving activities within the semiconductor technology sector. The European Union has set ambitious objectives to significantly strengthen its current not so strong position in this market. To achieve this, the EU has committed to substantial investments in this sector and aims to coordinate major projects across Europe. With significant contributions from the Czech EU Presidency, the key document ChipAct was approved, featuring a group of major projects in IPCEI ME/CT (the Important Projects of Common European Interest). This lecture aims to provide a concise overview of how national entities are involved in this new strategy and, through a specific example, demonstrate what the Czech Republic can contribute to the collective European semiconductor initiative.

Milan Semmler

Milan Semmler, a graduate of the Faculty of Nuclear Sciences and Physical Engineering at the Czech Technical University in Prague, Department of Dosimetry and Application of Ionizing Radiation. After his studies, he worked at the Nuclear Research Institute in the field of neutron radiography. After 1989, he moved to the private sector and co-founded CHEMCOMEX PRAHA a.s., a company that has implemented dozens of projects at nuclear power plants. He led the development team for the primary monitoring system of VVER power plants. After 2000, he focused on the modernization of radionuclide irradiators for the treatment of cancer patients produced at UJP PRAHA a.s., where he still serves as a board member responsible for the company’s research and development activities. He was appointed Vice President for Radiation-Resistant Microelectronics in the Czech National Semiconductor Cluster.

Progressive methods of driving permanent magnet synchronous motors (PMSM) with advanced algorithms and features

Speaker: **Ondřej Holý** *(STMicroelectronics, Czech Republic)*

Field Oriented Control (FOC) is a very well-known technique used to drive permanent magnet synchronous motors (PMSM) for many years. It has been adapted by many developers and is used widely across many different applications with PMSM. Some of the applications require dedicated features, such as a control of start-up from zero speed, Start-up On-The-Fly (OTF), Maximum Torque Per Ampere (MTPA), Discontinuous PWM (DPWM) and many others. Those are not coming by default with the FOC technique but require advanced algorithms and techniques.
In this workshop I will explain exclusively various advanced algorithms and techniques enabling such a features, including presentation of tools and resources that will ease their evaluation and implementation into own application use case. Benefits, resources constrains, and limits will be covered as well.

Ondřej Holý

Ondřej Holý joined STMicroelectronics in 2014 as a Microcontroller Support Application Engineer. Before joining STMicroelectronics, Ondřej worked as a hardware and software development engineer using a wide range of microcontrollers including STM32. Ondrej is responsible for supporting clients in the EMEA region, providing guidance and training on STM32 and Motor Control.

Assessing Computation Efficiency in Embedded Systems

Speaker: **Martin Daněk** *(daiteq, Czech Republic)*

Compared to commercial use, embedded systems for use in space have to consider additional design criteria, out of which perhaps the most important are radiation tolerance and power efficiency. The talk will present a number of metrics that can be used for assessment of computation efficiency, covering implementations ranging from dedicated hardware accelerators to custom processor instructions.

Single-cycle RISC-V processor microarchitecture design and implementation

Jan Medek, Ondrej Golasowski, Michal Stepanovsky
Faculty of Information Technology, CTU in Prague
Thakurova 9, 160 00 Prague 6, Czech Republic
{medekja5, golasond, stepami9}@fit.cvut.cz

Supervisor: *Michal Stepanovsky*

Abstract

The paper introduces a synthesizable single-cycle RISC-V RV32I processor microarchitecture, and its deployment as a soft-core processor on FPGA. The described microarchitecture emphasizes simplicity, making it suitable for teaching computer architecture courses or for use in simple embedded systems. The microarchitecture is described in Verilog Hardware Description Language. As the timing analysis shows, our synthesized soft-core processor on a Basys 3 FPGA development board (Artix-7) can operate correctly at 50 MHz, i.e. providing the execution speed of 50 MIPS (this is sufficient for many practical applications).

***Keywords*— RISC-V instruction set, Single-cycle microarchitecture, Soft-core processor, FPGA**

I. Introduction

Even though, in general, the processor itself is a highly complex electronic circuit, it externally offers a set of instructions (more specifically an Instruction Set Architecture, or ISA) that it can execute. There are different ISAs, with the most prevalent being x64 [1] (utilized in desktop computers, servers, etc.) and ARM [2] (employed in mobile phones and embedded devices). Nevertheless, both x64 and ARM are proprietary. RISC-V is an open-source alternative [3], allowing industry and academia to design and produce their own chips without incurring licensing fees. Moreover, the modularity of RISC-V allows for greater design flexibility. For these reasons, we focused on RISC-V ISA in this paper.

Because the microarchitecture design space is huge, we can encounter many different microarchitectures. These microarchitectures vary in the number of implemented instructions and their design complexity. There are, for example, single-cycle microarchitectures [4], pipelined microarchitectures with three [5], five [5], six [6], [7] or twelve [8] pipeline stages, in-order [7] or out-of-order [8], [9] superscalar microarchitectures, and multi-core microarchitectures [8], [10]–[12]. Design complexity also determines the areas of their use, from simple or advanced embedded systems [13], [14] to high-performance multi-core computer systems [8]. Because our main motivation is to present a simple microarchitecture, which can be used as a teaching example for computer architecture courses, we decided to design and implement a single-cycle microarchitecture. This microachitecture is described in the following sections.

II. The microarchitecture design

We focused on the RV32I, a base integer instruction set mandatory for every 32-bit RISC-V compatible processor [15]. We implemented all RV32I instructions, except `fence`, environment call `ecall` and the breakpoint `ebreak`, since we consider only a single CPU core and no operating system support in our design.

Table I summarizes all the instructions considered in our design. RV32I ISA defines four basic instruction formats (R, I, S and U-type) and two additional variants (B and J-type) based on the different handling of immediate operands. As shown in Figure 1, all R-type instructions manipulate two source registers (rs1 and rs2) and one destination (rd) register. All I-type instructions use one source register (rs1), one immediate operand (imm) and destination (rd) register. S/B-type instructions have three sources (rs1, rs2, and imm) and no destination register. And finally, U/J-type instructions have one immediate operand as a source and one destination register (rd).

Figure 2 represents a datapath of our microarchitecture. The datapath consists of several components, such as a program counter (PCReg), multiplexers (Mx), instruction memory (instrMem), general-purpose register file (GPR), arithmetic-logic unit (ALU), data memory (dataMem), immediate operand decoder (immDecoder) and others. All these components are interconnected in such a way that an arbitrary instruction from Table I can be executed in a single clock cycle.

A. Implementation in HDL

The proposed microarchitecture as presented in Figure 2 was implemented in Verilog HDL (1364-2005 IEEE Standard) [16] and is available at [17] under the folder `rtl/components`. The description of our microarchitecture was put into several files, as illustrated in Table II. In addition to the design as presented in Figure 2, we also implemented a simple support for a general-purpose input/output (GPIO). This allows communication with other input/output devices through the GPIO interface.

TABLE I
LIST OF IMPLEMENTED INSTRUCTIONS. RS1 AND RS2 REPRESENT THE CONTENT OF SOURCE REGISTERS; RD REPRESENTS THE CONTENT OF DESTINATION REGISTER; THE SYMBOLS $\oplus$, $\otimes$ AND $\odot$ ARE USED TO INDICATE VARIOUS OPERATORS, NAMELY $\oplus$ STANDS FOR +, &, $|, <<, <, <, >>>, >>, -$ AND THE ^ OPERATOR (MATCHING THE ORDERING OF LISTED INSTRUCTIONS), SIMILARLY $\otimes$ STANDS FOR +, &, $|, <<, <, <, >>>, >>$ AND THE ^ OPERATOR, AND FINALLY, $\odot$ STANDS FOR ==, $\geq$, $\geq$, <, < AND THE != OPERATOR. ALL INSTRUCTIONS SUFFIXED WITH "U" (E.G. `sltu`, `sltiu`, `lbu`, ETC.) ASSUME UNSIGNED INTEGERS AS OPERANDS (INSTEAD OF 2'S COMPLEMENT). THE NOTATION M[] IS USED TO INDICATE DATA MEMORY CONTENT. THE MEANING OF ALL LISTED INSTRUCTIONS IS GIVEN IN [15] IN DETAIL.

Instruction mnemonic	Type	Simplified description in Verilog
`add, and, or, sll, slt, sltu, sra, srl, sub, xor`	R	rd = rs1 $\oplus$ rs2;
`addi, andi, ori, slli, slti, sltiu, srai, srli, xori`	I	rd = rs1 $\otimes$ imm;
`jalr`	I	rd = PC+4; PC = rs1+imm;
`lb, lbu, lh, lhu, lw`	I	rd = M[rs1+imm];
`sb, sh, sw`	S	M[rs1+imm] = rs2;
`beq, bge, bgeu, blt, bltu, bne`	B	if(rs1$\odot$rs2) PC=PC+{imm,1'b0};
`auipc`	U	rd = PC+{imm,12'b0};
`lui`	U	rd = {imm,12'b0};
`jal`	J	rd = PC+4; PC = PC+{imm,1'b0};

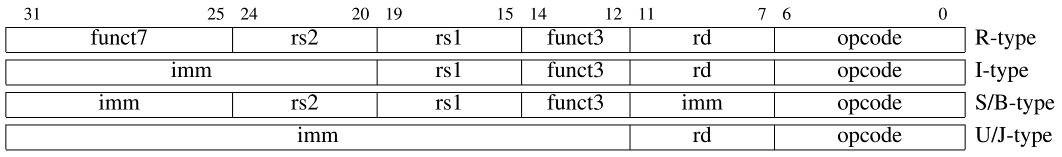


Fig. 1. RISC-V instruction formats. All instructions start with the opcode encoded using a 7-bit field in bits [6:0]. The remaining fields of the instruction differ depending on the instruction type. If the instruction contains a destination register (rd), its number is always encoded at the same position in all formats, i.e. in bits [11:7]. Similarly, if the instruction contains source registers rs1 and rs2, their numbers are always encoded in bits [19:15] and [24:20], respectively. We should note that in this Figure, rs1, rs2 and rd indicate where the corresponding register numbers are encoded, not the content of those registers.

The internal parts of the CPU (control unit, ALU, multiplexers, etc.) are described within the `cpu.v` file (and related modules/files), whereas the instruction memory, data memory, address decoder and GPIO stand outside the CPU. Their mutual inter-connection is described in the `top.v` file and illustrated in Figure 3, which is the target for simulation. This Figure actually represents a simple computing system able to execute a program stored in instruction memory, and able to communicate with data memory and the external world through GPIO. The target for synthesis is the `VESPTop.v` file which instantiates the `top.v` file and connects it's reset signal to reset synchronizer described in the file `synchronizer.v` (not discussed in this article). Then, the PLL (Phased-Locked Loop: usually used for shifting signal's phase or multiplying clock signals by rational number) template from Xilinx is used to divide the main frequency. This template is tool specific and can be replaced with any other module that configures the desired frequency by the user.

TABLE II
VERILOG SOURCE FILES USED TO DESCRIBE OUR DESIGN AS SHOWN IN FIGURE 2. FILES ARE LISTED IN ALPHABETICAL ORDER.

File	Description	In Figure 2 used for
`addressDecoder.v`	Address decoder	not shown (see Figure 3)
`alu.v`	Arithmetic-logic unit	ALU
`controller.v`	Control unit	Control signals
`cpu.v`	Mutual inter-connection of individual components to form a CPU; and defines multiplexers, program counter, adders and shift units	Inter-connections, Mx1-10, PCReg, Adder1-2, Sh1-2
`dataMemory.v`	Data memory	dataMem
`extend.v`	Parameterized sign/zero extension unit	ext8, ext16
`gpio.v`	General-purpose input/output	not shown (see Figure 3)
`immDecoder.v`	Immediate operand decoder	immDecoder
`instrMemory.v`	Instruction memory	instrMem
`registerFile32.v`	GPR file containg 32 registers	GPR
`top.v`	Simulation target: inter-connection of CPU with instruction and data memory	Inter-connections
`VESPTop.v`	Synthesis target: inter-connection of top.v with reset synchronizer and PLL used to divide the main frequency	not shown in Figure 2

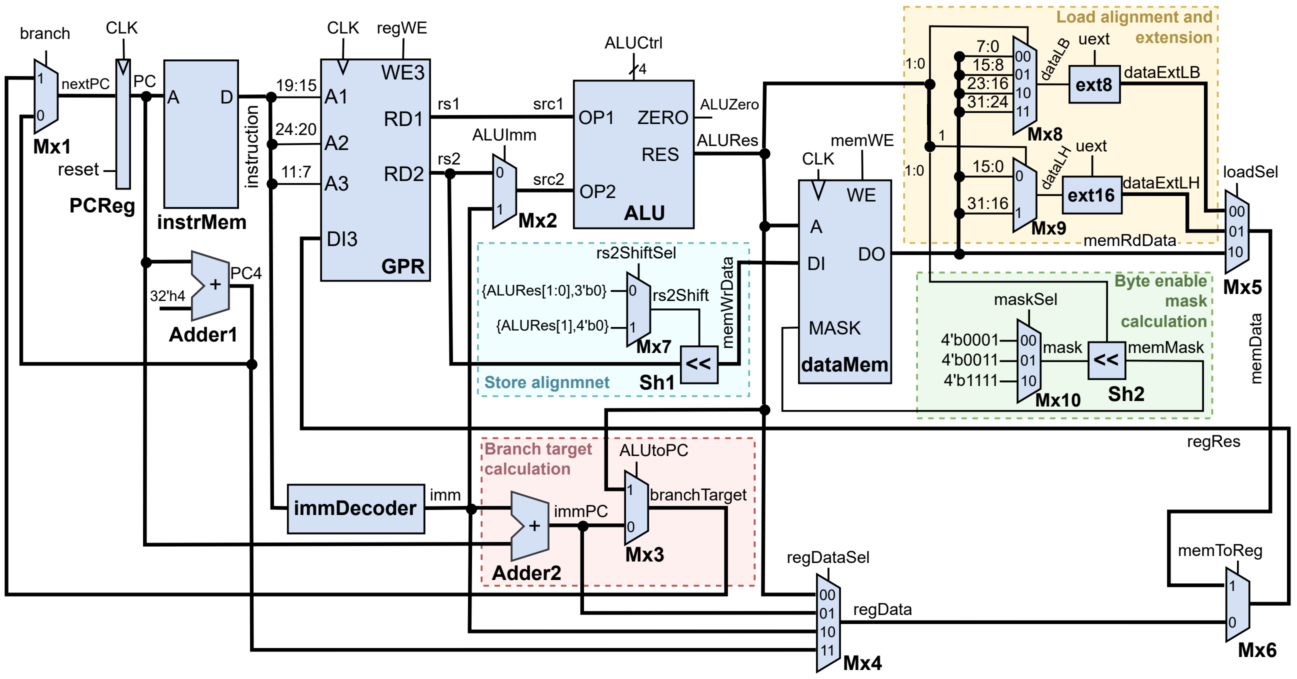


Fig. 2. Datapath of our microarchitecture

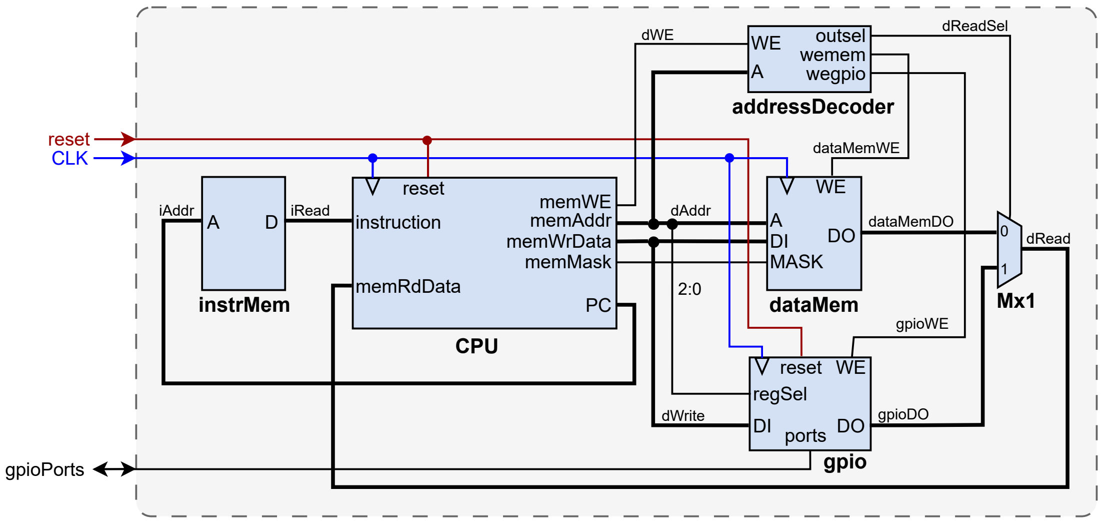


Fig. 3. Top-level datapath of CPU connected with instruction and data memory and with GPIO. This schematic corresponds to the Verilog code as given in the file `top.v` when SPLIT_MEMORY is defined (see Table II).

B. *Official RISC-V tests*

In order to confirm the functionality of our design, we used official unit tests for RISC-V processors [18]. They contain test programs written in assembly language that test whether the processor conforms to the ISA definition. We created the testbench module `riscvTopTest.v` to run all unit tests and to confirm whether these tests passed or not. In order to run these tests, the user has to execute the following command: `python make.py test` assuming that the user cloned our repository from [17]. We should note that these tests expect a common instruction and data memory (thus, the 4th line in `rtl/components/top.v` has to remain commented, and so the SPLIT_MEMORY is not defined).

III. Testing the designed soft-core processor on FPGA

To further verify and analyse the proposed microarchitecture, we synthesized and implemented our design into an FPGA, more specifically, into a Basys 3 FPGA development board (Xilinx Artix-7 xc7a35tcpg236-1). Resource utilization and timing analysis were evaluated in Xilinx Vivado v2018.3 and are presented in Table III and Table IV, respectively. Note that the resource utilization of the proposed processor is less than 6 %, thus leaving the room for various RISC-V extensions (e.g. support for floating-point numbers). Since the critical path delay is estimated to be 17.2 ns and the Basys3 board includes a 100 MHz oscillator, we defined a timing constraint for the clk signal to be 20 ns, so the oscillator can be used after dividing its frequency by a factor of 2. As shown in Table IV, the worst slack is 2.156 ns, thus the processor will operate correctly at 50 MHz frequency allowing to execute 50 MIPS (million instructions per second). We should note that the acceptable clock frequency depends on various parameters (used FPGA board, pin assignments, or even the size of the instruction and data memories).

TABLE III
FPGA usage

Site Type	Used	Fixed	Available	Util%
Slice LUTs	1143	0	20800	5.50
LUT as Logic	839	0	20800	4.03
LUT as Memory	304	0	9600	3.17
LUT as Distributed RAM	304	0		
LUT as Shift Register	0	0		
Slice Registers	66	0	41600	0.16
Register as Flip Flop	66	0	41600	0.16
Register as Latch	0	0	41600	0.00
F7 Muxes	168	0	16300	1.03
F8 Muxes	64	0	8150	0.79

TABLE IV
Timing analysis for the critical path

Slack (MET)	2.156ns	(required time - arrival time)
Requirement	20.000ns	(clk rise@20.000ns - clk rise@0.000ns)
Data Path Delay	17.224ns	(logic 4.544ns (26.383%) route 12.680ns (73.617%))
Logic Levels	18	
Clock Path Skew	-0.099ns	
Clock Uncertainty	0.089ns	

IV. Conclusion

In this article, we have designed a simple microarchitecture implementing the RV32I instruction set from the RISC-V family. The design of this microarchitecture is described in Verilog Hardware Description Language (1364-2005 IEEE Standard). The microarchitecture is implemented as single-cycle, and thus the clock cycle has the same length for every instruction. However, as the timing analysis shows, this is acceptable for this small instruction set. More specifically, our synthesized soft-core processor on a Basys 3 FPGA development board (Artix-7) can operate correctly at 50 MHz, i.e. providing the execution speed of 50 MIPS (this is sufficient for many practical applications). Our simulation simulation results confirm that the design of our microarchitecture is in accordance with the RV32I RISC-V ISA definition. Moreover, the described design is published as open-source, and can therefore be used directly or adapted to the actual needs.

Acknowledgment

This work was supported by the Student Summer Research Program 2023 of FIT CTU in Prague and SGS23/208/OHK3/3T/18 (”Design, programming and verification of intelligent embedded systems”).

References

[1] "Intel 64 and ia-32 architectures software developer's manual volume 1: Basic architecture." https://cdrdv2.intel.com/v1/dl/getContent/671436. [Accessed 16-11-2023].

[2] "Arm architecture reference manual for a-profile architecture." https://developer.arm.com/documentation/ddi0487/latest/. [Accessed 16-11-2023].

[3] "About RISC-V; RISC-V International – riscv.org." https://riscv.org/about/. [Accessed 16-11-2023].

[4] D. K. Dennis, A. Priyam, S. S. Virk, S. Agrawal, T. Sharma, A. Mondal, and K. C. Ray, "Single cycle RISC-V micro architecture processor and its FPGA prototype," in *2017 7th International Symposium on Embedded Computing and System Design (ISED)*, pp. 1–5, 2017.

[5] T. Zheng, G. Cai, and Z. Huang, "A soft RISC-V processor IP with high-performance and low-resource consumption for FPGA," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2538–2541, 2022.

[6] J. Mach, L. Kohútka, and P. Čičák, "A new RISC-V CPU for safety-critical systems," in *2023 12th Mediterranean Conference on Embedded Computing (MECO)*, pp. 1–4, 2023.

[7] K. Patsidis, D. Konstantinou, C. Nicopoulos, and G. Dimitrakopoulos, "A low-cost synthesizable RISC-V dual-issue processor core leveraging the compressed instruction set extension," *Microprocessors and Microsystems*, vol. 61, pp. 1–10, 2018.

[8] C. Chen, X. Xiang, C. Liu, Y. Shang, R. Guo, D. Liu, Y. Lu, Z. Hao, J. Luo, Z. Chen, C. Li, Y. Pu, J. Meng, X. Yan, Y. Xie, and X. Qi, "Xuantie-910: A commercial multi-core 12-stage pipeline out-of-order 64-bit high performance RISC-V processor with vector extension : Industrial product," in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pp. 52–64, 2020.

[9] C. Celio, D. A. Patterson, and K. Asanović, "The berkeley out-of-order machine (BOOM): An industry-competitive, synthesizable, parameterized RISC-V processor," Tech. Rep. UCB/EECS-2015-167, EECS Department, University of California, Berkeley, Jun 2015.

[10] A. Kamaleldin and D. Göhringer, "Agiler: An adaptive heterogeneous tile-based many-core architecture for RISC-V processors," *IEEE Access*, vol. 10, pp. 43895–43913, 2022.

[11] A. Kamaleldin, S. Hesham, and D. Göhringer, "Towards a modular RISC-V based many-core architecture for FPGA accelerators," *IEEE Access*, vol. 8, pp. 148812–148826, 2020.

[12] S. Savas, Z. Ul-Abdin, and T. Nordström, "A framework to generate domain-specific manycore architectures from dataflow programs," *Microprocessors and Microsystems*, vol. 72, p. 102908, 2020.

[13] B. Sá, J. Martins, and S. Pinto, "A first look at RISC-V virtualization from an embedded systems perspective," *IEEE Transactions on Computers*, vol. 71, no. 9, pp. 2177–2190, 2022.

[14] W.-P. Kiat, K.-M. Mok, W.-K. Lee, H.-G. Goh, and R. Achar, "An energy efficient FPGA partial reconfiguration based micro-architectural technique for iot applications," *Microprocessors and Microsystems*, vol. 73, p. 102966, 2020.

[15] "The RISC-V instruction set manual, volume I: User-level ISA, document version 20191213." Editors Andrew Waterman and Krste Asanovic, RISC-V Foundation, https://riscv.org/technical/specifications/, Dec 2019.

[16] "IEEE standard for verilog hardware description language." IEEE Std 1364-2005 (Revision of IEEE Std 1364-2001), doi:10.1109/IEEESTD.2006.99495, 2006.

[17] O. Golasowski and J. Medek, "VESP-alpha: RISC-V based student processor for embedded applications." https://github.com/andreondra/vesp-alpha/tree/article-Medek-version, 2023.

[18] "riscv-tests." https://github.com/riscv-software-src/riscv-tests. [Accessed 28-05-2024].

Can Optimization Principles from Synchronous Adders be Applied to Asynchronous (QDI) Ones?

Oliver Haschke
Faculty of Informatics at TU Wien
A-1040 Wien, Treitlstrasse 1-3/E191-02
e1636305@student.tuwien.ac.at

Supervisors: *Florian Huemer, Andreas Steininger*

Abstract

While synchronous designs are bound to constraining the clock to the worst case, asynchronous design techniques can provide so-called "average case" performance by flexibly adapting their timing to operating conditions and even input data. However, in order to maintain global co-ordination in the absence of a global clock, they need to employ local handshaking and obey specific protocols, which causes performance and area penalties.

In this paper, we use the example of a binary adder to investigate the properties of asynchronous techniques relative to the popular synchronous style. In particular, we analyze whether optimizations from the synchronous domain, set out to reduce the (worst case) carry propagation path, are of any benefit in an asynchronous implementation, specifically a "quasi delay-insensitive" one. Our results show that the advantage of being able to consider average case rather than worst case for performance is significant, but the QDI style's need for a 4-phase handshake nullifies this gain. The carry-chain optimizations of the Kogge-Stone adder, while effective for the worst case, even turn out counter-productive for the average case.

***Keywords*— carry chain, Kogge-Stone Adder, Ripple-Carry Adder, QDI Adder**

I. Introduction

The theoretical performance gain of asynchronous logic is immediately obvious for an adder, as the worst-case performance that all synchronous logic components are bounded by is instead shifted to an average-case performance which is thus faster on average. So, while in a synchronous design the fixed clock period must accommodate the longest feasible carry propagation, no matter the current input, an asynchronous adder can provide the output, once the specific calculation is indicated as complete. For the conventional ripple-carry adder (RCA) this can be concretized by the findings of Von Neumann [1], which note that given uniformly distributed random operands the average length of the longest carry chain is proportional to $log(n)$ where n is the width of the adder in bits. This must be contrasted against the theoretical worst-case performance of the RCA which has, depending on the exact configuration, a longest possible carry chain either of length n or $n-1$.

Of course, asynchronous logic not only comes with benefits but also suffers from drawbacks. In particular less common and thus less optimized logic components, such as the Muller C-gate or threshold gates must be utilized to meet the requirements that a given asynchronous design style imposes. Furthermore, industrial-grade design tools do not support asynchronous logic, which also hampers the creation of optimized asynchronous circuits. And finally, to maintain appropriate flow control, asynchronous designs have to provide handshakes that adhere to special protocols and, in case of the quasi delay-insensitive (QDI) timing model [6] considered in our study, special data encodings. These tend to degrade performance and increase area. So the first question we want to answer (actually revisit) here is whether the benefit of aligning the speed of operation to the actual delays rather than the worst case can offset those performance penalties.

Given the importance of adder circuits as primitives in many complex functions, countless advancements have been made for binary adders to improve their performance. And due to the dominant effect of the carry path on the propagation delay, many such techniques focus on reducing the longest possible carry chain, which would then allow the adder to be operated at higher clock frequencies. However, asynchronous implementations already purport average-case performance. Since the improvements for synchronous adder designs were focused on improving the worst-case in particular, it remains to be seen if these same techniques can also improve an asynchronous design, which already achieves average-case performance. Furthermore, it remains to be seen whether the cost of these improvements, judged for example by the additional required area, is worthwhile when compared with the asynchronous RCA. These are our further research questions. We will base our investigations and comparisons on simulation results and design tool reports for actually synthesized adder circuits for various bit widths.

II. Background and Related Work

A. Preliminaries

Before diving into the specific contents, let us introduce the main relevant concepts for this subject matter: The **critical path** is the longest possible signal path through the circuit and in the case of synchronous logic its length serves as a lower bound for the clock period, given that any signal must be able to traverse the critical path within one clock period. In contrast, an asynchronous circuit has a data-dependent critical path.

Unlike the ripple-carry adder, which simply calculates each carry as part of the single-bit binary addition and then passes this carry on to the next single-bit adder block, i.e. a full adder block, modern designs achieve some level of parallelism for the calculation of the sum of the binary addition by calculating the carries ahead of time and as much as possible in parallel. The carry-lookahead adder is the most basic one of these, but it still suffers from internal fan-out problems and unfavorable scaling towards larger n. Thus to relieve the previous logic or buffer stage and to achieve better parallelism the concept of the **group carries** is introduced. The group carry signals $G_{i:j}$ and $P_{i:j}$ represent that a group of bit-positions from i down to j will generate a carry at its end, the i^{th} position, or propagate the incoming carry, c_{j-1} to the end, c_i, respectively. These group carries are defined as follows:

$$G_{i:j} = G_{i:k} \wedge P_{i:k} \wedge G_{k-1:j} \qquad\qquad P_{i:j} = P_{i:k} \wedge P_{k-1:j} \tag{1}$$

with the base case

$$G_{i:i} = G_i = g_i \qquad\qquad P_{i:i} = P_i = p_i \tag{2}$$

As the papers concerned with such adders show, smaller groups of bits can be calculated in parallel and the groups can then be combined to form the carry signals for larger groups of bits. This is usually accomplished using a tree structure, hence the adders utilizing this techniques are commonly referred to as parallel-prefix-tree adders (PPTA).

B. Asynchronous Logic

As already mentioned, asynchronous logic employs local handshakes for co-ordination rather than a global clock. This handshaking is generally based on a **request** and an **acknowledge**. The request signals to the data receiver (further on called **sink**) that new data is available for processing, whereas the acknowledge notifies the sender (further called data **source**) that the sink is ready for new data. The acknowledge is usually a transition of a dedicated ack signal (wire). The request can either be conveyed over an explicit signal (req) as well, or it can also be implicit in the data, for example by utilizing an encoding for the data from which the sink can infer that the transmitted data is complete. We will elaborate more on this later on.

The handshaking can be realized in a 2-phase or a 4-phase manner. For the **2-phase protocols**, also called non-return-to-zero (NRZ), each transition on req signals the availability of new data to the sink. The following transition on ack then notifies the source that the sink is ready for new data. In contrast, the **4-phase protocols**, sometimes referred to as return-to-zero (RZ), are based on the states of the two signals and not on the transitions. This means the for example a HI on req indicates the readiness of the data, and a H on ack indicates that the data has been consumed. Naturally, both signals then have to return to LO, to initialize themselves for the next data cycle. This phase of returning to the idle states is referred to as **null phase**.

For the **bundled data (BD)** design style [8] req is explicit and "bundled" with the data: A binary data word is sent to the sink without any special extra encoding, and the accompanying req indicates its validity. Naturally, the request must not arrive at the sink before data is actually valid, which creates a (relative) timing condition. Since data often undergoes some processing on the path to the sink, the request must be artificially delayed (by insertion of extra gates, which may consider data dependence) to obtain so-called delay matching. **Delay-insensitive (DI)** designs use data encoding to restrict this timing assumption to the minimum, namely positive and bounded delays for all wires and gates [9]. The class of circuits, however, that can be built on this paradigm in practice, is extremely small [10]. As a compromise, the **Quasi Delay Insensitive (QDI)** timing model has been introduced. It extends DI by introducing the concept of isochronic forks [10], [11]: For some selected wire forks within the design it must be guaranteed that the signal transitions occur at the same time (in practice, with very small skew) at all endpoints of the fork. All other delays, in particular the gate delays, are completely unrestricted, as they would be in a DI design. The introduction of this only relaxation to the otherwise delay insensitive designs allows that arbitrary circuits can be constructed.

Both DI and QDI designs encode the request signal into the transmitted data, for which a variety of different codes is available. Probably the simplest and most popular one is the **dual-rail encoding**, where a single logic signal is mapped to two physical wires, referred to as rails. The logic signal x, which can be either $true$ or $false$ is mapped to the dual-rail signal $x = (x.T, x.F)$ where $x.T$ being HI represents that x is $true$ and $x.F$ being HI represents x is $false$. Such a dual-rail encoding can thus be realized as an RZ encoding with (0,0) being the idle state and (1,1) forbidden.

The ultimate purpose of the data encoding is to allow the sink (actually a so called completion detector near the sink) to detect whether all bits of a data word have arrived. This process is referred to as the **completion detection (CD)**. Notice that, with the choice of a appropriate delay-insensitive code, this eliminates the race condition between req and data as seen with the BD design. Still, however, enforcing strict separation between data phases and null phases in all combinational and buffer stages, as implied by the 4-phase protocol, may become costly in terms of performance and area. Instead, some designs have opted to use auxiliary $done$ signals indicating completeness of (internal) parts of the function, whose combination can then safely indicate completion without the need for introducing costly and slow completion detectors directly into the data path.

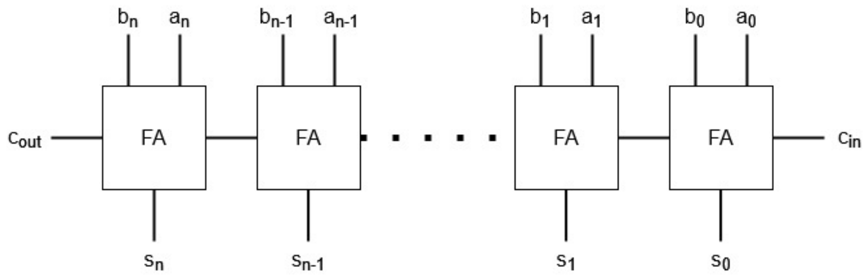


Fig. 1. Basic block diagram of a RCA with carry-in

In the remainder of this paper we will focus on the popular 4-phase protocol, using the QDI model with dual-rail encoding. This is also known as Null-Convention Logic (NCL). As laid out above, we will make use of auxiliary $done$ signals, which specifically brings us to the so called **NCLX** approach (NCL with eXplicit completion detection) [7].

C. Related Work

The 1994 paper "Performance Comparison of Asynchronous Adders" [2] by Franklin et. al. compares various well known adder designs, namely the Ripple Carry Adder (RCA), Conditional Sum Adder (CSA), Carry Lookahead Adder (CLA), Carry Skip Adder (SKP) and Carry Select Adder (SEL), with the Completion Detection Conditional Sum Adder (CDA). The CDA is a CSA with additional detection logic that indicates the availability of the true sum at each level to often avoid the full tree delay, $O(log_2(n))$. As with all asynchronous circuits the delay of the CDA is variable and data dependent and its mean can be as small as $O(log_2(log_2(n)))$. The adders are implemented using a dual-rail encoding and differential cascode voltage switch logic (DCVSL), and are considered in the context of a pipeline of an asynchronous RISC processor. Overall the CDA had the best performance, but it was not considered for the evaluation of the overall pipelined processor, as it was the only adder with some additional overhead. Finally an overview for the 6 adder designs is given for 32 bit and 64 bit addition, and a further overview for the pipelined processor is given for the 5 adders, sans CDA. Ultimately the SEL outperformed the other designs in regards to throughput of the asynchronous processor.

In his 1996 paper "An Evaluation of Asynchronous Addition" [3] David Kinniment evaluates an adder with self-timing aspects that utilizes a dual-rail CD for the carries, although the completion status of each pair of rails is collected in an n-input AND gate rather than a Muller C-gate. He compares this adder (which he refers to as ASY) with the known Conditional Sum Adder (CSA) and a Simple Parallel Adder (SPA), which is essentially an RCA, with respect to the total number of gates needed to implement each adder, as well as the number of gate delays incurred by each adder. For this purpose the AND, OR, NAND and NOR gates are considered equivalent, and an AOI structure is considered 80% of 2 levels of NAND gates. The adders are compared with each other, and also with the results from other papers, based both on random data and data based on usage in a 32 bit ALU. The performance is also evaluated in the context of a micropipeline where it is assumed that the other pipeline stage has a matching delay with a small possible deviation. He concludes that the conditions in which the ASY can outperform the CSA are very limited, and do not hold up to the scrutiny of real data in a real application.

The paper "A Comparison of Quasi-Delay-Insensitive Asynchronous Adder Designs corresponding to Return-to-Zero and Return-to-One Handshaking" [4] by P. Balasubramanian and C. Dang gives a comparison between RCAs implemented in QDI. In particular the RCAs were implemented using QDI Full Adders (FA). The FAs themselves were differentiated based on the chosen implementation style, namely Strong Indication, Weak Indication or Early Output. Additionally both a 4-phase Return to Zero (RTZ) and 4-phase Return to One (RTO) implementation were considered. The results indicated that the RTO implementations outperformed the respective RTZ ones in regards to area and power usage, as well as latency and cycle time.

III. Implementation

A. Considered Adder Architectures

For a given width of n bits the **ripple-carry adder (RCA)** is implemented by simply chaining n full-adder (FA) blocks together, connecting the $carry-out$ of the previous FA to the $carry-in$ of the subsequent FA (Figure 1). This is arguably the simplest adder architecture, so we will investigate it further as a baseline, also because most asynchronous adder implementations so far are RCAs. With a critical path

$$t_{crit} = n \cdot t_{FA} \tag{3}$$

(with t_{FA} being the delay of an FA block) directly proportional to n (i.e., $O(n)$) it has bad worst-case performance, and as already laid out, we expect the most benefit when implementing it in asynchronous logic.

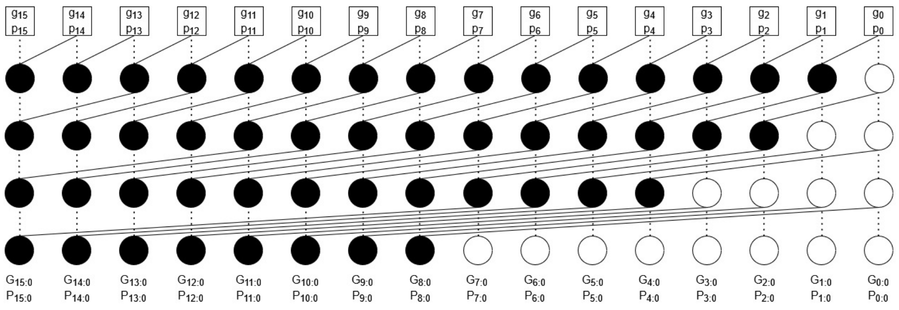


Fig. 2. Prefix tree of a 16 bit KSA

The **Kogge-Stone adder (KSA)** is based on a general algorithm for solving recurrence problems introduced by Kogge and Stone in [5]. This algorithm resembles the blueprint for a tree structure that calculates the group carries in a parallel manner, just like specified in Equations (1) and (2). An example of this prefix tree is given in Figure 2, where the filled, black nodes represent a processing node which implements the logic according to the previously mentioned equation, and the empty, white nodes represent a simple throughput where the output is equal to the input.

The KSA comes with some important benefits that make it worthwhile for our comparison, namely it achieves a constant fan-out of 2 at every node and it achieves very good performance due to its high degree of parallelism and its low logic depth. The logic depth for the prefix tree of a KSA is $log_2(n)$.

Equation (4) gives the critical path for the KSA where t_{GC} is the delay of one processing node, t_{pg} is the delay of the logic block that calculates the initial g_i and p_i, and t_{sum} gives the delay of the logic block that calculates the sum:

$$t_{crit} = log_2(n) \cdot t_{GC} + t_{pg} + t_{sum} \tag{4}$$

These two selected adders were implemented in VHDL and – using the Synopsys Design Compiler (DC) – mapped to an open 45nm cell library which provides the required basic logic gates (AND, OR, XOR) and function blocks (FA, MUX). For the asynchronous versions the NCLX design style was used. To allow covering the parameter space (adder type – synchronous/asynchronous – bitwidth) in a time-efficient way the synthesis and simulation process was automated using a shell-script.

B. Synchronous Implementation

The synchronous designs were implemented in a straightforward manner by instantiating and interconnecting the cells provided in the aforementioned library. No additional optimization by hand was done and the optimizations done automatically by Synopsys DC were entirely disabled. This approach was done for two main reasons. First it was discovered early in the implementation process that the DC, when not otherwise configured, ”optimizes” the more complex adder designs into logically equivalent combinations of prefix tree and ripple-carry logic, which when simulated performed significantly worse than the simple ripple-carry adder. In addition, the asynchronous designs were not supposed to be optimized by the tool anyways, since we would require the logic to be exactly what we specified, so as to guarantee the QDI properties of the design (that DC is not aware of). This required specifying a so called ”don't_touch” attribute for all cells and nets within the design. Furthermore the boundary optimization which optimizes across sub-design boundaries was turned off as well.

C. Asynchronous Implementation

To implement the NCLX adder designs it was necessary to first create NCLX versions of the required logic gates and function blocks from the available cells of the library. Additionally an implementation of the Muller C-gate from the library cells was required.

As previously mentioned, a possible advantage of the NCLX design style is the avoidance of C-gates on the data path. Thus the NCLX versions of the logic gates and function blocks were also created without a CD or $done$ signal. This essentially makes the NCLX versions of the gates just dual-rail implementations of the respective logic gates. For example the AND gate shown in Figure 3 is essentially just one logic gate per output-rail which produces the required logic function for this rail: The NCLX AND gate has one OR gate to assert the $false$ rail when either of the input $false$ rails is asserted, and one AND gate which asserts the $true$ rail when both input $true$ rails are asserted. The NCLX OR gate works analogously. The NCLX version of the XOR gate is slightly more complicated as it requires 4 AND gates to check for each of the possible input signal combination and then 2 OR gates to turn the resulting 1-hot-code into appropriate activation of the gate's $true$ and $false$ output rails. This approach allows us to forgo unnecessary, duplicate CDs for example for the input signals, since

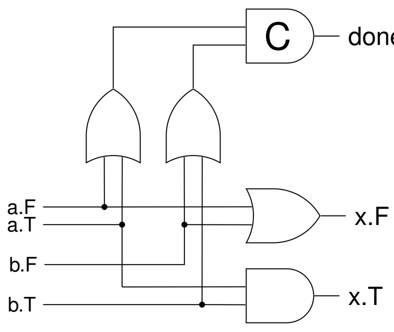


Fig. 3. NCLX AND gate with explicit input CD

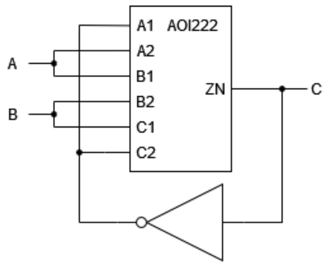


Fig. 4. Muller C-gate built from AOI structure

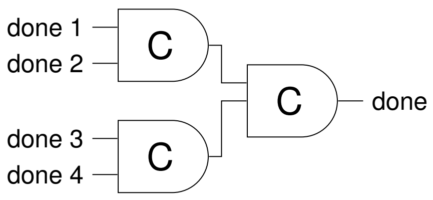


Fig. 5. Muller C-gate tree

rather than each NCLX gate individually checking whether the input signal has arrived, we can check a given signal just once and (relying on the isochrony of wire forks) assume that it has arrived at all of its endpoints.

The slightly more complex function blocks, namely the FA and MUX (that had been directly available as library cells for the synchronous implementation) also needed to be explicitly created from the NCLX logic gates. The Muller C-gate was implemented using an AOI222 cell from the library, as well as an inverter cell that connects the AOI's output pin ZN back to the input C to create a storage loop that exhibits the desired hysteresis behavior, see Figure 4. Furthermore, an additional buffer cell was needed to decouple this C input node from the C-gate's output.

The individual C-gates of the CD are then combined into a tree structure, as shown in Figure 5, and the individual $done$ signals then combined into a final, single $done$ signal for the combinational logic. As the figure shows two internal $done$ signals can feed into one C-gate, however, the intermediate results of these C-gates have to be combined in an additional C-gate as well. In general we can say that $n-1$ C-gates are necessary for combining n $done$ signals. The same would also be true and perhaps even more evident if we were to combine the $done$ signals in a C-gate chain, where at every stage the respective C-gate combines the results of the previous stage with one new signal to form the new result. It should be noted here that for the tree structure and for $n : n > 1 \land n \neq 2i : i \in \mathbb{N}$ there are multiple ways of constructing the tree structure, however, neither the required number of C-gates nor the maximum depth and thus the longest path through the tree depend on the specific way that the tree is constructed. The advantage to using a tree structure is naturally that the longest path through the structure is much shorter as it has logarithmic rather than linear scaling. As Figure 3 shows, the actual logic behind the CDs and the required C-gate structure are rather simple. For each dual-rail signal one OR gate is needed.

IV. Simulation Results

A. Simulation Setup

All implemented adder design variations were simulated using ModelSim, with the simulation runs being automated through the scripts described in the previous section. Beyond systematic selection of synchronous vs NCLX implementation, RCA vs KSA, and bit width of 8/16/32/64, the operands were also automatically varied. For the latter it could be specified whether the operands of the addition are selected randomly (based on a uniform distribution), exhaustively, or as a counter, that simply counts from 0 to the maximum for the given width. Simulation of the 8 bit wide adders supports all 3 modes, whereas 16 bit only supports the random and the counter mode, and the wider widths only support the random mode. This limitation in the supported modes comes from the total value range that the respective number of bits represents. For example exhaustively testing the 16 bit adders amounts to 2^{32} combinations, counting from 0 to the maximum for 32 bits comes to the same number of additions, and it is simply not feasible to run simulation for multiple billions of additions.

B. Simulation Run

A simulation run of a given adder design and simulation configuration involves embedding the adder in a generic testbench which, based on the simulation mode, selects the operands and applies them to the adder. The testbench then reports the operands, as well as the start time, end time, and duration of the addition in a CSV file. For the duration only the time until the correct result is present at the outputs is counted. This is done in a loop for a specified number of iterations. The random values for the operands are generated using the functionality provided by the "RandomPkg" from OSVVM. This also guarantees that we get the same random operands for the different designs, so that they are properly comparable. For the asynchronous adders the testbench also has to apply the null phase after each addition, however, the null phase was not initially counted as part of the addition in regards to the reported time. Rather for the asynchronous adders, in addition to the duration of the addition, the time until $done$ transitioned to high, indicating that the data-phase is now complete, as well as the time until $done$ transitioned to low, indicating that the null-phase is complete, were also reported in the CSV file. This allows us to observe the data-path through the designs and the CD separately and judge both the data-path only and the complete data-phase when comparing the designs against each other. Additionally the synchronous adders were also reset after each performed addition, since especially for the smaller two bit-widths it was not uncommon to have two consecutive pairs of operands which produce the same result, resulting in a duration of 0 for that particular simulation iteration.

C. Area and Timing Analysis

Unlike the delay values obtained by simulation, the area usage of the adders is directly reported by DC using the ”report_area” command. The area report contains a listing of the number of ports, nets, and cells, as well as a further breakdown of the types of cells, e.g. combinational or sequential. Additionally the report also contains the total area usage of the design, as well as a breakdown based on the type of area, e.g. combinational, non-combinational or interconnect area. The report itself does not contain a unit for the area value, rather the unit has to be taken from the library to which the design is mapped. For the open library we used the area is given in square micro meters [μm^2]. Furthermore the used library has an area value of 0 defined for the wire load models. Thus the area report cannot calculate a total interconnect area for the design, instead the interconnect area and by extension the total area of the design remain as ”undefined”. Due to this circumstance we will instead use the ”total cell area” as the principal value for all of the comparisons. However, in the grand scheme of things this should not cause any significant difference, as we would expect the total interconnect area to be roughly proportional to the aforementioned reported numbers, i.e. number of ports or number of nets.

Similarly the timing, or more specifically the maximum timing path in the design, which is the critical path, can be reported directly by DC using the command ”report_timing”. The generated report contains a listing of the points in the design along this unconstrained maximum path, the increment that the given point adds and the cumulative length, measured in time, that the path has from the start to that point. The report ends with the ”data arrival time” which for our unconstrained, maximum path is the time when the data arrives at the output, when starting from the input at time 0. Like the unit for the area the unit for the time values is also defined by the used library, in this case the timing is given in nanoseconds [ns]. Furthermore it is possible to set the number of reported digits to control the output format for the reported values. This has no influence on the precision of the calculation, which is always done with the maximum possible precision that the operating system supports for floating point arithmetic. While the maximum timing path is not as important for the asynchronous designs, since they will be compared based on their average delay, as calculated from the simulation results, it is very important for the synchronous designs, since they will be compared primarily based on this reported value.

D. Data Presentation

The results of the simulation are stored as a CSV file containing one row for each executed addition. These CSV files are then loaded into a Jupyter Notebook, a Python based tool with a Web-browser-based user interface, where they are processed. This processing involves an evaluation of the statistical parameters of the given simulation series. In particular, the average and the maximum for any given adder are of importance. Furthermore, the synchronous adders have the delay of their reported maximum timing path, which is the critical path of the design, as described in Section II-A also added to the processed results. These results are plotted as box plots, since this allows to show all of the relevant data in a very compact form. The box plots have the average, maximum and the reported worst-case path overlaid, to make comparison easier. Furthermore the number of nets a given adder architecture uses is also reported, and then visualized in a bar graph.

V. Discussion of Results

A. Performance

Let us start with a comparison between the two synchronous implementations, in order to establish relative performance gain of the more complex KSA architecture against the RCA. In Figure 6 we show the 16 bit variant only, due to space limitations; the trends for the other bit widths are similar. As can be seen in the box plots, the average delays are very similar. Based on the already stated observations this an expected outcome, since the primary focus of the improvements of the modern adder designs lies in reducing the critical path, since in a synchronous design the worst case is relevant for clock frequency and hence performance. Indeed, we can observe that the KSA has significantly reduced worst-case paths when compared to the RCA. Interestingly, we can also observe that the KSA has its delay values very tightly grouped around the mean while they are widely spread for the RCA. This is may seem obvious for the delay values which are larger than the mean. When the mean and the maximum or in this specific case the worst-case delay are closer together then the values in between are naturally more tightly grouped. However, for the KSA this is also true for the delays below the mean. Since the prefix tree has to be traversed in any case this means that a baseline delay is introduced even for the simplest additions. Thus in specific cases the RCA can produce faster additions than the KSA.

When we now examine the delay results of the asynchronous adders, as shown in Figure 7, then we can see that the results are almost the same. As for these results the CD was not considered, this shows us that expanding the conventional single-rail

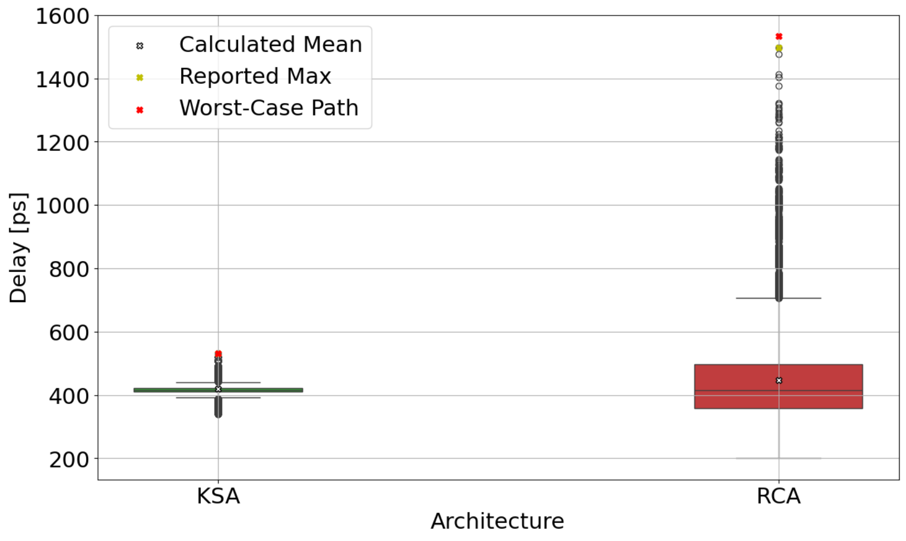


Fig. 6. Delay results of the synchronous 16 bit adders

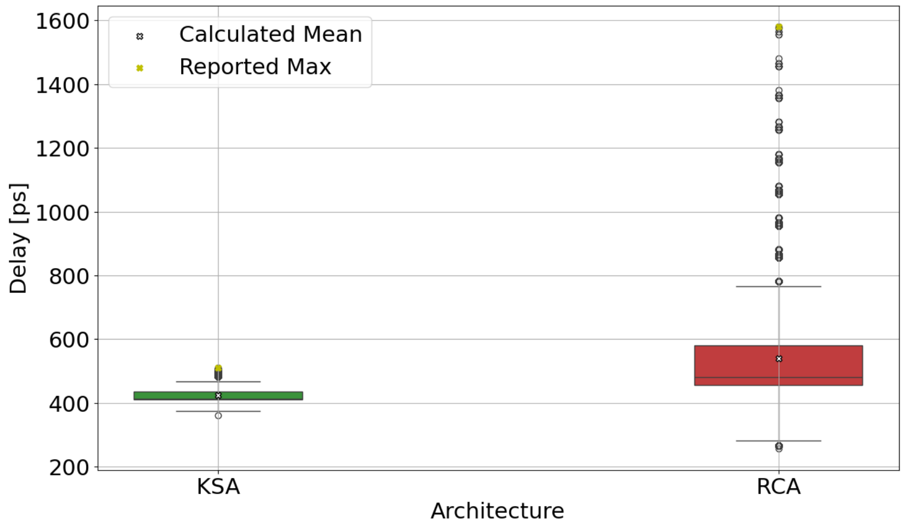


Fig. 7. Delay results of the QDI 16 bit adders for the data-phase, without CD

logic to the dual-rail equivalent has, as expected, no noticeable impact on performance. Of course, the main drawback of this design style so far is that singular standard library cells are replaced with multiple such cells to implement the logic function. As mentioned in Section III-C some of the cells such as XOR or MUX require multiple cells to implement. Because these cells are not only in parallel, as is the case for the AND and OR for example, certain paths through the designs became slightly longer when implemented in this way. This leads to the observable differences in the results between Figure 6 and Figure 7. However, for the comparison between RCS and KSA we can make the same general observations as in the synchronous case.

So in the next step we will now also consider the CD and observe how this changes the results regarding performance. The results are shown in Figure 8 and it is immediately obvious that the CD has a big impact on the overall performance of the adders. While we can observe that the delay values still show the same grouping patterns, in regards to how tightly the delays are grouped for each architecture, we can now also see that the CD takes a significant amount of time to produce the *done* signal. Since the CD has a different level of complexity for RCA and KSA, we can see that the results are not merely shifted upwards by a constant amount, instead the more complex KSA takes significantly longer to produce its *done* signal than the RCA. This is of course because the size of the CD increases with the number of intermediate signals in the design for which their arrival must be detected. As a consequence we can also observe that (at least for this bit width) the RCA, in spite of

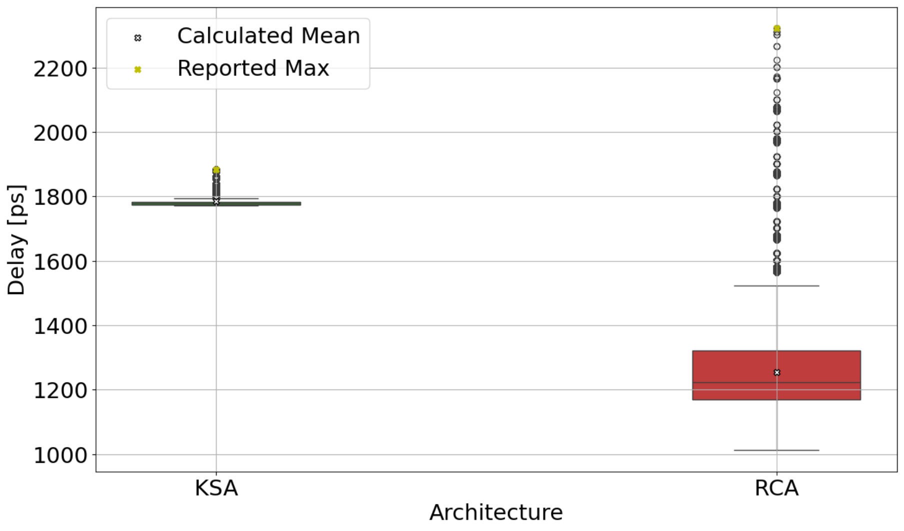


Fig. 8. Delay results of the QDI 16 bit adders for the data-phase, with CD

still having the longer maximum delay, also has the shorter average delay, which is what we care about most in regards to the performance of asynchronous adders. So, ironically, increasing the level of parallelism (as the KSA does) makes the CD more complex and ultimately degrades performance.

Beyond the influence of the CD, another important performance aspect in 4-phase QDI designs is the need for a null phase. To include that in our picture, Figure 9 shows the results of the null-phase (only) with the CD also considered. The first thing that we can take notice of is that the delay values are much more tightly grouped, which is of course due to the fact that we are supplying the same input values (idle) for each null-phase. So while the starting point of each null-phase might be different, both the supplied inputs and desired result are the same. We can also see that it takes the KSA longer to execute the null-phase than the data-phase.

For the overall performance we can make the following conclusions, as evidenced by Figure 10: As the average delay is significantly smaller than the worst case delay, an asynchronous adder implementation has much potential for performance benefits. Even the more complex gate-level implementation does not change this picture much. However, even though the CDs are more cleverly arranged in NCLX than in a plain NCL implementation, they cause an appreciable performance drop, aggravated by the need for a null phase. This is true for both, RCA and KSA. On top of that, the optimizations made for the KSA improve the worst case delay at the cost of higher average case delay, which makes the KSA perform even worse in an asynchronous implementation. However, towards larger bit widths (like 64 in the figure), the worst case path for the synchronous RCA scales even worse than the overheads for the CD in the QDI KSA. At this point the QDI RCA becomes the second best choice behind the synchronous KSA which is the clear winner.

B. Area

Figure 11 shows the areas for the two adder types in their synchronous and QDI implementations, as reported by the synthesis tool (note the logarithmic scale). For the asynchronous variants, the expected penalty for DI encoding and CD is clearly visible, most prominently for the KSA. Also, their scaling towards larger bit widths is unfavorable. It is, however, also interesting to see that the synchronous KSA does not scale well either.

C. Nets and Fan-out

Figure 12 shows the number of nets for the two adder types in their synchronous and QDI implementations, as reported by the synthesis tool (again in logarithmic scale). The picture is essentially the same as with the area. The absolute numbers shown in the graph should be taken with a bit of care, as our test setup introduced several extra signals, but the relations and trends are still representative.

VI. Conclusion

We have analyzed whether the performance gains obtained by optimizing the critical path of a synchronous RCA, like in a KSA, can be harvested in an asynchronous implementation as well. To this end we have compared an RCA and a KSA in both, synchronous and asynchronous (QDI) implementation by means of extensive simulation. Not surprisingly, the synthesis

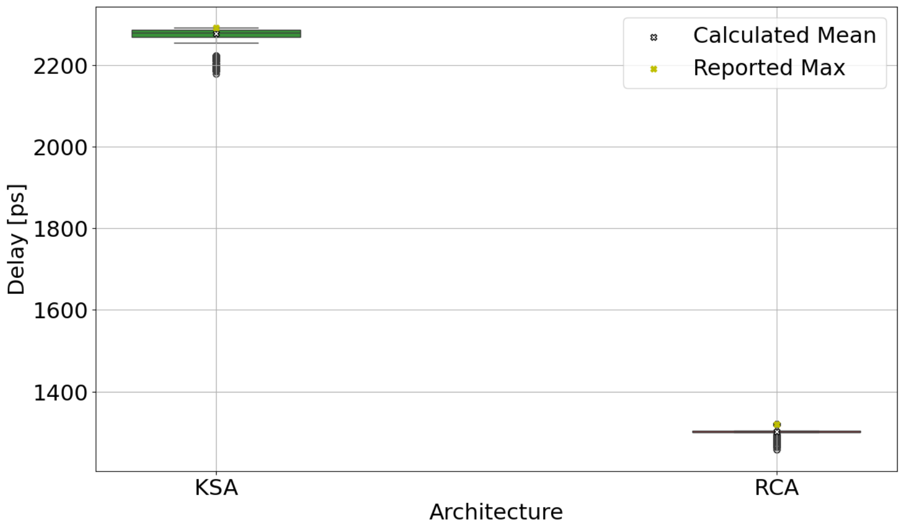


Fig. 9. Delay results of the QDI 16 bit adders for the null-phase, with CD

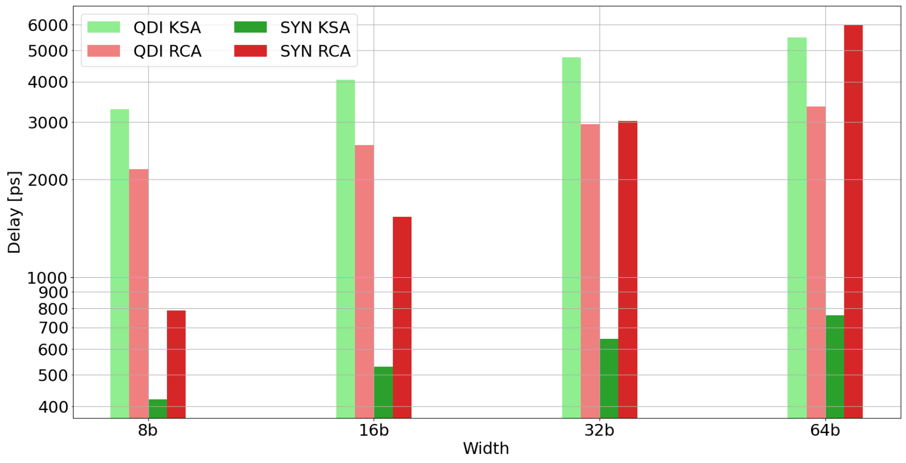


Fig. 10. Comparison of the synchronous worst-case vs. the mean cycle-time of the QDI adders

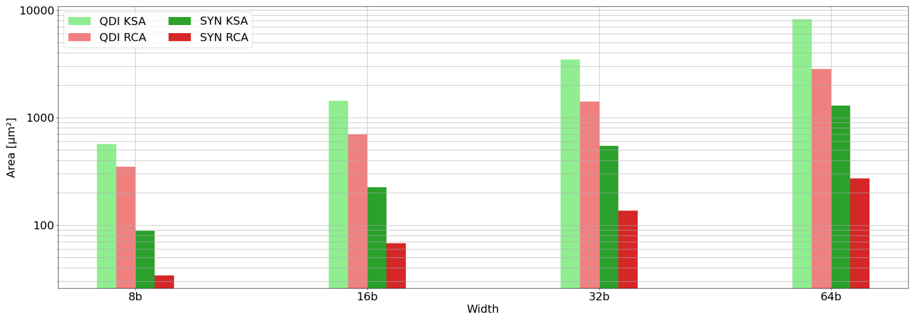


Fig. 11. Area usage of adders

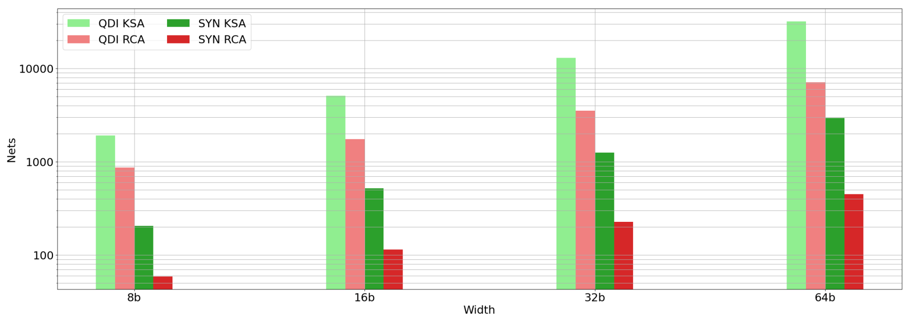


Fig. 12. Number of nets

reports clearly indicated overheads for area and number of nets for the QDI implementations, as these require extra logic for dual-rail encoding of data and for completion detection.

With respect to performance, we could observe the clear advantage of completion detection, yielding a performance related to the average case, rather than being bound to the worst case, as in the synchronous case. However, as our results showed, this benefit is nullified by the 4-phase protocol that demands for an (unproductive) null phase, as well as the delays incurred by the completion detection (even in case of the more beneficial NCLX). This becomes less pronounced for higher bit widths (starting at 32bit) where the asynchronous RCA beats the synchronous one, but in any case the synchronous KSA is the clear winner, while its asynchronous variant cannot benefit from the structural optimizations, as these are targeted to the worst case path only and essentially deteriorate the average-case timing.

A. Future Work

While the asynchronous adders can not compete with the synchronous KSA when viewed purely as standalone components, these adders would be embedded within a pipeline in a real world scenario. The research done in this paper indicates the CD as the clear bottleneck in regards to the performance of the asynchronous adders. Therefore a primary optimization goal should be the minimization of the CD, and to this end there are some clear advantages that can be gained once the adders are embedded in a pipeline. Most obviously the input CD of the adder can be shared with the CD of the preceding storage bank of the pipeline, thus effectively removing it from the combinational block, i.e., the adder. Additionally the succeeding storage bank also has its own CD and the *done*-signal of the adder need only be available once the latching of the results is done for this storage bank, as according to its CD. Thus the two CDs can partially run in parallel. How this affects the overall performance of the complete pipeline, when compared to a synchronous pipeline merits further exploration.

Paper Origin

This paper originates from a Master thesis at TU Wien which will be published as such.

References

[1] Von Neumann and A.H. Taub. Collected Works. Number Bd. 1. Pergamon Press, 1963.
[2] M. A. Franklin and T. Pan, "Performance comparison of asynchronous adders," Proceedings of 1994 IEEE Symposium on Advanced Research in Asynchronous Circuits and Systems, Salt Lake City, UT, USA, 1994, pp. 117-125.
[3] D. J. Kinniment, "An evaluation of asynchronous addition," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 4, no. 1, pp. 137-140, March 1996.
[4] P. Balasubramanian and C. Dang, "A comparison of quasi-delay-insensitive asynchronous adder designs corresponding to return-to-zero and return-to-one handshaking," 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), Boston, MA, USA, 2017, pp. 1192-1195.
[5] Peter M. Kogge and Harold S. Stone. A parallel algorithm for the efficient solution of a general class of recurrence equations. IEEE Transactions on Computers, C-22(8):786–793, 1973.
[6] R. Manohar and Y. Moses, "The Eventual C-Element Theorem for Delay-Insensitive Asynchronous Circuits," 2017 23rd IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), San Diego, CA, USA, 2017, pp. 102-109, doi: 10.1109/ASYNC.2017.15.
[7] A. Kondratyev and K. Lwin. Design of Asynchronous Circuits by Synchronous CAD Tools. In Proceedings 2002 Design Automation Conference (IEEE Cat. No. 02CH37324), pages 411–414, June 2002.
[8] I. E. Sutherland. 1989. Micropipelines. Commun. ACM 32, 6 (June 1989), 720–738.
[9] Jan L.A. van de Snepscheut. Trace Theory and VLSI Design. LNCS 200, Springer-Verlag, 1985
[10] Martin, A.J. (1990). The Limitations to Delay-Insensitivity in Asynchronous Circuits. In: Feijen, W.H.J., van Gasteren, A.J.M., Gries, D., Misra, J. (eds) Beauty Is Our Business. Texts and Monographs in Computer Science. Springer, New York, NY.
[11] Martin, A.J. Compiling communicating processes into delay-insensitive VLSI circuits. Distrib Comput 1, 226–234 (1986).

Analysis of Statistical Distribution Changes of Input Features in Network Traffic Classification Domain

Lukáš Jančička, Josef Koumar, Dominik Soukup
Czech Technical University in Prague, Prague, Czech Republic
16000 Thákurova 9, Prague 6, Czech Republic
{janciluk, koumajos, soukudom}@fit.cvut.cz

Tomáš Čejka
CESNET, a.l.e.
Prague, Czech Republic
cejkat@cesnet.cz

Abstract

This study investigates the evolving landscape of network traffic monitoring, which is crucial for maintaining computer network services and security. Traditional methods like Deep Packet Inspection (DPI) face challenges due to increased privacy protection through encryption, prompting a shift towards statistical-based detection using Machine Learning (ML). On the other hand, ML struggles with long-term evaluation due to various distribution changes. This study focuses on the CESNET-TLS-Year22 dataset, derived from one year of TLS network traffic on the CESNET2 backbone. Described research explores the behavior of modern protocols in real-world scenarios and their impact on dataset quality. The main result of our analysis is the identification of the Weekend phenomenon in network traffic classification that is generally overlooked during ML model training.

We analysed the statistical properties of the dataset for each class, and we find out interesting concept drift that appears on weekdays and holidays. Thus, this behavior appears on the days when most people do not work and students are not in universities and dormitories. We call this behavior a Weekend phenomenon in the network traffic classification domain. The CESNET-TLS-Year22 dataset was captured on the ISP network CESNET2, which mainly contains networks of universities, academic institutions, and dormitories for students. Therefore, the Weekend phenomenon may be greater than in other ISP networks. However, on the corporate networks, the Weekend phenomenon may be the same or bigger than the CESNET2 network.

If we combine the results of KS tests with the feature importance of model, which was trained on the whole first week, we find out that most of the features that are important for the model have different distribution on weekdays and weekends, for example, PPI Size and Direction packets for the first 10 packets, and bytes and packets in both directions. Moreover, the features that have the same distribution on weekdays and weekends ended with small feature importance. Thus, we assume that the model is forced to learn more from the features that have different distribution properties on the weekend. The provided results also bring consideration for training period and ML model deployment, since, for example, a training period of one day with a single ML model can be affected by the Weekend phenomenon. More details are part of the future work.

Keywords— Traffic Analysis, Traffic classification, TLS, Deep learning, Computer network, Concept drift

Acknowledgement

This research was partially funded by the Ministry of Interior of the Czech Republic in project “Flow-Based Encrypted Traffic Analysis” (VJ02010024), and by the Ministry of Education, Youth and Sports of the Czech Republic in project “e-Infrastructure CZ” (LM2023054). Computational resources were provided by the e-INFRA CZ project (ID:90254), supported by the Ministry of Education, Youth and Sports of the Czech Republic.

Paper Origin

Previously published and presented at IEEE/IFIP Network Operations and Management Symposium (NOMS) 2024

TCI: A system for distributed network monitoring, troubleshooting and dataset creation.

Dominik Soukup, Jaroslav Pešek
Czech Technical University in Prague & CESNET a.l.e.
Prague, Czech Republic
{soukudom, pesekja8}@fit.cvut.cz

Lukáš Hejcman, David Beneš, Tomáš Čejka
CESNET a.l.e.
Prague, Czech Republic
{hejcman, benes, cejka}@cesnet.cz

Abstract

Network traffic monitoring is a very complex task that requires a combination of multiple tools and teams. Very often, detected events must be validated and confirmed, or ongoing detection needs additional detailed data from full packets. All these activities must be done automatically concerning data privacy. This is why we propose a solution in the form of Traffic Capture Infrastructure (TCI), a single system for network traffic capture, investigation, and dataset creation, even in high-speed provider networks. This system is designed to be a single solution for all mentioned tasks, even at large-scale networks, including user management, audit logs, and application programming interface (API). This system has already been deployed on the infrastructure of the Czech Education and Scientific Network (CESNET) organization. CESNET is a developer and operator of national e-infrastructure for science, research, development, and education in the Czech Republic. The system has been used successfully in multiple papers, publications, and open-source datasets. This paper presents the architecture, main functions, recommendations, and lessons learnt from full packet monitoring in today's networks. Lastly, we prove the value of this system with several publications that have used our system to create their underlying dataset and network traffic investigation.

Introduced system was created as a single solution for network traffic capture, investigation, and dataset creation. The role of TCI is to enhance current network monitoring tools with full packet capture operations, which is needed in many activities. Thanks to the described functions and experience, TCI simplifies network operations for researchers, SOC analysts, and network operators. TCI is a production-ready system that is deployed at the CESNET2 network. Moreover, it is also publicly available [1]. Additionally, we show an overview of the structure and architecture of the system and explain its main functionality, together with identified recommendations for full packet capture monitoring even in a high-speed environment. Finally, we give an overview of the main use cases of the system and prove its usefulness using a list of publications that used the TCI system during its development.

***Keywords*— network monitoring, datasets, network troubleshooting**

Acknowledgement

This research was funded by the Ministry of the Interior of the Czech Republic, grant No. VJ02010024: Flow-Based Encrypted Traffic Analysis, and by the MEYS of the Czech Republic, grant No. LM2023054: e-Infrastructure, and by internal grant of CTU in Prague, grant No. SGS23/207/OHK3/3T/18.

Paper Origin

This paper has beeen previously published and presented at IEEE/IFIP Network Operations and Management Symposium (NOMS) 2024.

[1]https://github.com/FETA-Project/TrafficCaptureInfrastructure/tree/main

Implementation of a Bump-in-the-Wire Security Gateway

Philipp Schwind, Tobias Frauenschläger and Prof. Dr. Jürgen Mottok
Laboratory for Safe and Secure Systems (LaS³)
Technical University of Applied Sciences
OTH-Regensburg
{philipp.schwind, tobias.frauenschlaeger, juergen.mottok}@oth-regensburg.de

Abstract

In the face of escalating cyber threats, the integration of Operational Technology (OT) with Information Technology (IT) systems presents heightened vulnerabilities to critical infrastructures. This paper focuses on the need for cybersecurity, showcasing the design and implementation of a security gateway to achieve bump-in-the-wire security between two units to be protected (assets). For that, each asset is assigned one security module that safeguards its communication paths. To ease bump-in-the-wire capability, the communication between the asset and the security-gateway must be stateless and operate on OSI layer 2. This is necessary to safeguard the whole communication bandwidth, regardless of the used protocols. For this objective, Linux-based packet sockets are used. An encrypted tunnel is established between two security gateways using the Datagram Transport Layer Security (DTLS) protocol. The results show that the implemented solution is capable of securing the asset's network traffic without the need of any modifications.

Index Terms

bump-in-the-wire security, Linux, VPN, DTLS

I. Introduction

Increasingly, our digital landscape is besieged by cyberattacks, a trend alarmingly highlighted in the 2018 Ponemon Institute study [1, 2]. This surge underscores an urgent demand for enhanced cybersecurity measures. In a world where digitalization permeates every facet of our lives, the urgency for robust cybersecurity measures has never been more pronounced. As early as 2003, the former German Federal Minister of Interior, Otto Schilly, captured the need for action in his statement: "There is no security without IT security" [3]. This statement is even more important today, as we increasingly rely on information technology in critical sectors such as energy, healthcare, finance, water and many more. The potential impact of cyber threats on these essential services escalates, posing not just economic risks but also threats to public safety and national security. According to the Microsoft Digital Defense Report from October 2021, the nature of cyber operations is complex and evolving [4]. As an example, the Russian hacker group NOBELIUM was behind the SolarWinds Orion breach, a highly sophisticated supply chain attack that compromised thousands of organizations worldwide by inserting malicious code into software updates [4, 5]. This example emphasizes the emergence of hackers perpetrating increasingly complex cyberattacks on critical infrastructure. Moreover, the ongoing digitalization is leading to an increasing convergence of Information Technology and Operational Technology (OT), heightening vulnerabilities as OT systems have been isolated from the public Internet in the past [6]. The current developments underline the increasing threats and the critical need for robust cybersecurity. In response to the cyber threat landscape, the German government issued the Kritische Infrastrukturen (KRITIS) law, mandating state-of-the-art IT security for operators of critical infrastructures [7, 8]. In this context, the Laboratory for Safe and Secure Systems (LaS³) undertakes the KRITIS³M project, aiming to implement various security gateways. This project seeks to cryptographically secure communication paths, enhancing the resilience of essential services against cyberattacks of any kind. The aim is to find software and hardware solutions that facilitate the dynamic exchange of cryptographic algorithms, enabling the continual use of state-of-the-art IT security. By using this pattern, security can be maintained against emerging technologies, such as quantum computing. This capability to quickly and dynamically adopt cryptographic methodologies is referred to as cryptographic agility [9].

Given the expanded connectivity between OT and IT systems, the attack surface has notably increased, highlighting the vulnerability of distributed systems to unauthorized access and potential man-in-the-middle attacks. Due to the difficult updatability of OT devices, it is researched in retrofitting these devices with state-of-the-art IT security, without necessitating any client-side modifications. The proposed solution involves the injection of two transparent security devices into a point-to-point connection between two assets. A cryptographic tunnel is established between these security modules, ensuring the integrity, authenticity, and confidentiality of the data. This method is known as bump-in-the-wire security [10, 11].

The secure tunnel is achieved by implementing a Virtual Private Network (VPN) application for the security gateway. To meet the criterion of requiring no modifications on the client side, the security module is designed to function seamlessly within the communication chain. Achieving this necessitates the gateway's operation at Open Systems Interconnection (OSI)

Layer 2, enabling it to handle the entire network traffic of an asset, irrespective of the packet's protocol. This work seeks to answer the research question: **How can a bump-in-the-wire security gateway be implemented that operates on Layer 2?**

To achieve this, the bump-in-the-wire security gateway meets the following requirements: One security gateway exclusively extends one asset and safeguards the data regardless of the protocol. The bump-in-the-wire security gateway can function as either a separate physical device or be deployed as a software service directly on the asset, and it must remain transparent to the asset. Inside the VPN tunnel, the data is cryptographically secured. Outside the VPN tunnel, the data remains unchanged, as if no security modules were in use, thereby eliminating the need for modifications on the asset side. This retrofit approach avoids the need to redesign already functional assets and offloads the security responsibilities to the security gateway. By incorporating features like secure VPN tunneling with DTLS, and a scalable, interface-patterned software architecture, a comprehensive blueprint for enhancing cybersecurity measures is provided.

This paper is structured as follows: Chapter II reviews the existing literature on bump-in-the-wire security gateways and implementation on VPN gateways. Chapter III provides an overview of theoretical foundations. Chapter IV outlines the design and implementation, explaining the concept of the implementation. Chapter V presents the implementation of the security gateway application. Chapter VI evaluates the implementation, elaborating the performance and functionality. Chapter VIII concludes the paper, summarizing the findings and outlining future research directions.

II. Related Work

Building on previous research, this work draws on contributions from the field of cybersecurity and network communications.

Bump-in-the-wire security by Jason Staggs and Sujeet Shenoi [10]: In the development of a layer 2 security gateway, the book provides a foundational approach to secure Supervisory Control and Data Acquisition (SCADA) systems. Their approach aligns with the NIST 2018 security framework, which emphasizes the vital necessity for robust cybersecurity defenses in OT systems [12]. Their research focused on a bump-in-the-wire solution that can be retrofitted on SCADA field devices to provide security functionality. The presented bump-in-the-wire solution serves as a fundamental approach for the security-gateway to retrofit devices of the OT with security features.

Protocol evaluation for VPN applications by Du meng [13]: This work presents a VPN solution, utilizing a UDP tunnel and OpenVPN's TAP interface[1]. The author chose UDP over TCP due to its superior performance in this use-case. The difference in performance occurs as soon as TCP packets are transmitted encapsulated in a TCP connection, which is referred to as TCP meltdown [14, 15].The reason for the lower throughput is attributed to both TCP flow controls. If a packet gets lost, both flow controls may request the same packet again, leading to multiple retransmissions of the same packet. This effect escalates in unreliable networks and results in a low average throughput.

The OpenVPN TAP interface was selected for its ability to facilitate the passage of any application-layer protocol through the network. The performance of the UDP-based VPN was thoroughly evaluated, showing significant improvements in network throughput and overall effectiveness in a real network environment [13].

DTLS performance evaluation by Gallenmüller Sebastian et al [16]: In the paper, a DTLS based VPN gateway was developed in order to determine the performance of generic DTLS enabled applications. The built multicore architecture handled encryption in a user-space program with performance and scalability in mind [16]. Additionally, a short comparison of DTLS to the UDP based crypto protocol WireGuard was given. It outlined the suitability of WireGuard for a lightweight and fast VPN solution. But as a drawback, they pointed out the lack of support for cipher suites, since WireGuard only supports the ChaCha20-Poly1305 cipher. In contrast to WireGuard, DTLS is superior for a crypto agile approach due to the rich support of cipher suites. Influenced from this paper, DTLS was selected for the implementation.

Influenced by these works, the bump-in-the-wire approach is used to retrofit OT devices. To implement a layer 2 bump-in-the-wire security gateway, a Linux packet socket in promiscuous mode is used. The cryptographic protection is achieved using the DTLS protocol, which is suitable for a crypto agile approach as well as for VPN applications. Following this chapter, the concept of the security gateway is explained.

III. Fundamentals

This chapter covers the foundational knowledge required to understand the implementation of our bump-in-the-wire security gateway. Section III-A explains the basic functionality of a VPN. Section III-B details the functionality and use-case of a packet socket and Section III-C of a Tap interface. Section III-D discusses the functionality of DTLS and how the protocol achieves the security objectives of authenticity, confidentiality, and integrity.

[1]https://github.com/OpenVPN/openvpn

A. Functionality of a VPN

A VPN creates a secure and encrypted connection between a VPN client and server over a less secure network, typically the internet [17]. It allows remote users and websites to communicate securely as if they were directly connected to a private network. The connection between the client and server is called the VPN tunnel. Within the tunnel, packets of the private network are encapsulated in a cryptographically protected packet. Tunneling protocols are used for the cryptographic protection and encapsulation mechanism. This is achieved through tunneling protocols and encryption. There are two primary types of VPNs: client-to-site VPNs and site-to-site VPNs.

Client-to-site VPNs connect individual users to a private network. This setup requires a VPN client on the user's device to establish a secure connection to the VPN server within the private network. This allows the user to access network resources as if they were physically present in the private network.

Site-to-site VPNs, on the other hand, connect entire networks to each other. This type of VPN requires a VPN server and client at both ends to establish a secure connection and is suitable for applications where both sites need to interact directly [18]. In the context of securing communication between two assets, this method is preferred because it treats both assets as equal communication partners, unlike the client-to-site approach. To implement a site-to-site VPN with two security gateways, each gateway's client connects to the other's VPN server. Once the VPN tunnel is established, both gateways can independently route packets bidirectionally between their private networks.

B. Functionality of a Linux Packet Socket

A Linux packet socket operates at OSI layer 2, and receives or transmits raw Ethernet frames [19]. This functionality is particularly useful for implementing network monitoring tools, custom protocol implementations, or a security gateway.

When a packet socket is created, it can be bound to a specific network interface and configured to capture packets based on certain criteria. For the bump-in-the-wire security gateway, the packet socket serves as the entry point to the VPN tunnel, meaning that all packets, regardless of the destination MAC address, should be received and forwarded to the tunnel. Therefore, promiscuous mode is enabled [20]. Similarly, when sending data through a packet socket, the application must construct the entire packet, including all necessary headers.

C. Functionality of a Tap Interface

A TAP interface is a virtual network kernel device that, similarly to the packet socket, operates at layer 2 [21]. Unlike packet sockets, which provide access to raw packets, TAP interfaces act as network interfaces that can be used by applications to send and receive Ethernet frames. This makes TAP interfaces particularly suitable for creating virtual network environments, VPNs, and other networking applications that require direct access to Ethernet frames.

The primary difference between a TAP interface and a packet socket is the abstraction level they provide. While packet sockets require applications to handle raw packets with all their headers, TAP interfaces allow applications to interact with the network stack as if they were using a standard Ethernet interface.

For the bump-in-the-wire security gateway, a TAP interface coexists next to the packet socket and can be used as an entry point to the VPN tunnel as well. By using a TAP interface, the VPN application can be deployed on the same device without the necessity of a separate physical device.

D. DTLS overview

As highlighted in Chapter II, datagram protocols are superior for VPN tunnels. DTLS adapts the robust security framework of Transport Layer Security (TLS) to the dynamic conditions of datagram networks, ensuring integrity, confidentiality, and authenticity [22]. The latest version DTLS 1.3 is used, which is specified in RFC 9147 [23].

Security goals: In the following, the basic functionality of DTLS and the implementation of the security goals authenticity, integrity, and confidentiality are explained:

Authenticity: To guarantee the authenticity of communication entities, a Public Key Infrastructure (PKI) is used. The server requires a valid entity certificate issued by a trusted Certificat Authority (CA), along with its corresponding private key. The client needs the CA's certificate chain to verify the server certificate. These certificates can be obtained from public CA's or internal CA systems [24]. The PKI itself is not within the scope of this work. Both gateways already have the necessary certificates. In KRITIS environments, adopting mutual authentication is essential, embodying the principles of zero trust architecture and ensuring both server and client authentication [25].

Integrity and confidentiality: Integrity, ensuring that data remains unchanged during transit, is maintained through a Message Authentication Code (MAC). Confidentiality, ensuring that data is inaccessible to unauthorized parties, is achieved through symmetric encryption. For both objectives, the Authenticated Encryption with Associated Data (AEAD) cipher AES GCM is used.

Unlike TLS, DTLS is providing communication privacy for datagram protocols, such as UDP. Therefore, DTLS must deal with peculiarities of the transport protocol, such as the unreliability.

Reliable DTLS handshake: The DTLS handshake is similar to the TLS handshake and involves a cryptographic negotiation in the beginning, where both entities exchange supported methods for the handshake. In this step, it is agreed on cipher suites, a key exchange algorithm such as the Diffie-Hellman (DH) algorithm and a signature algorithm [23]. Then a secret key is negotiated. The handshake requires a reliable connection, which is not included in UDP. Therefore, DTLS built its own transmission control which is applied during the handshake. It uses the following techniques: **Reordering**: Sequence numbers are used to ensure messages are processed in the correct order. If a message sequence number does not match the expected one, the message is queued until all previous messages are received [23]. **Message Size**: DTLS messages are fragmented over multiple DTLS records to overcome the UDP packet size limitation due to Maximum Transmission Unit (MTU). Fragment offset and length fields are used to reassemble the original message [26]. For this project, the third-party library WolfSSL[2] is used to integrate the DTLS protocol into the security gateway.

The next chapter details the concept of the security gateway, which implements a VPN and uses DTLS as transport layer security protocol to protect the data within the tunnel.

IV. Concept

The application implementation includes two major aspects: bump-in-the-wire capability and a cryptographically protected VPN tunnel.

Bump-in-the-wire capability requires the security modules to be transparent to their clients. To achieve this, a stateless Ethernet connection between the asset and the gateway is established, allowing all network traffic, regardless of protocol, to be received. **Cryptographic protection** is ensured through DTLS. In alignment with zero-trust architecture principles, mutual authentication is applied during the handshake [27]. Figure 1 illustrates the gateway's conceptual framework, employing the red/black principle for visual distinction [28]. Areas marked in red indicate unprotected communication where data is transmitted in plain text, denoted as the trustzone. Consequently, the gateway and the asset must be located within a physically secure environment to minimize the risk of unauthorized access. The assets communicate with each other via security modules, with each security module acting as both a VPN client and server. By using both services, the site-to-site VPN functionality is implemented, which concept was introduced in chapter III-A.

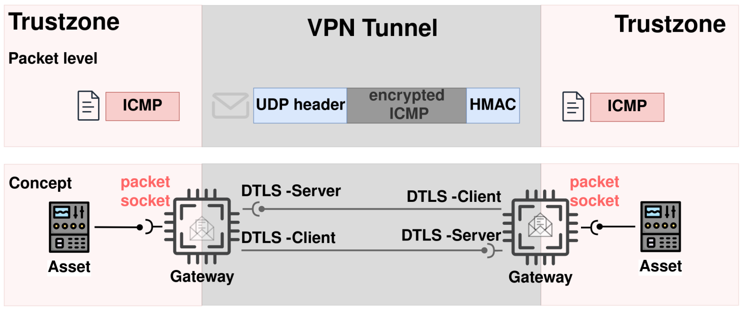


Figure 1: Two assets are connected to each other via the secure tunnel using two separate security-modules.

The trust zone interface employs a Linux packet socket in promiscuous mode to receive OSI layer 2 packets even if not being the true recipient. This is necessary, since the packets are addressed to an endpoint within the counterpart trustzone and not to the security-gateway. This methodology enables bump-in-the-wire capability. Figure 1 shows an asset sending an Internet Control Message Protocol (ICMP) packet to the other asset using the security gateways. When the gateway receives the packet at the asset interface, the packet is encrypted and sent as payload within a UDP packet to the counterpart gateway, where the payload is decrypted and forwarded to the connected trust zone. This process introduces an overhead because, in addition to the raw packet, new MAC, IP, UDP, and DTLS headers are added.

Figure 2 visualizes the packet length in the trustzone compared to the packet length in the tunnel, both captured on layer 2. The packets were generated using the `ping` command with variable packet sizes. The packets are then captured in the tunnel and in the trustzone using Wireshark[3].

[2]https://github.com/wolfSSL/wolfssl
[3]https://www.wireshark.org/download.html

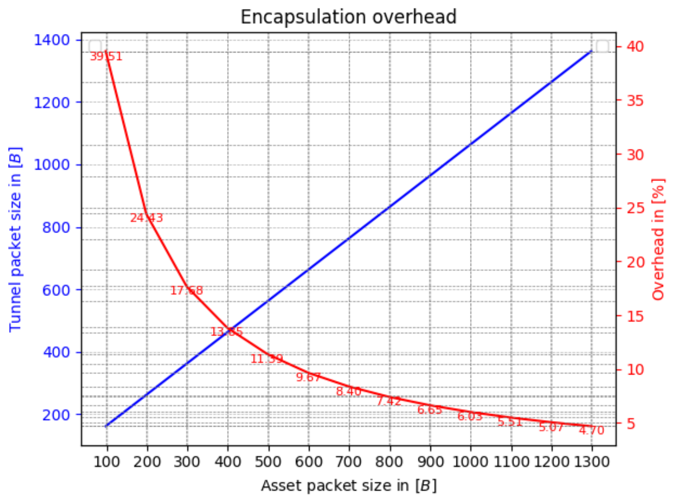


Figure 2: Visualization of the relative packet overhead introduced by tunneling

The relative overhead depends on the payload size. For instance, the standard ping message is 98 bytes long, but the corresponding tunnel packet is 162 bytes. Due to the packet overhead, there is a limitation on the maximum packet length, as the MTU is limited to 1500 bytes for Ethernet interfaces [29]. To ensure that the packet size in the tunnel does not exceed the 1500-byte limit, the gateway's MTU on the trustzone is set to 1300 bytes. In this work, the MTU for the assets is also set to the limit of 1300 bytes to ensure that packets can be sent in full. This approach contradicts the bump-in-the-wire approach, as it requires client-side changes to be made. However, this problem is to be resolved as part of future work. For example, the determination of the Path Maximum Transmission Unit (PMTU) or packet fragmentation can be used for this purpose [30]. The subsequent steps specify both communication paths: asset-to-gateway and gateway-to-gateway.

Asset to gateway: This communication path is a short-distance and stateless connection. Both the asset and the gateway must be in the same network, since the communication is layer 2 based.

Gateway to gateway: This communication path is inherently stateful, requiring a coordinated handshake between a DTLS client and a server, as described in Chapter III-D.

This work implements a single point-to-point connection between two security modules. To make this concept work in a multi-security gateway infrastructure, a service must be implemented that determines the target security gateway for a given packet. Therefore, the security modules could implement a service similar to the Address Resolution Protocol (ARP) protocol, as illustrated in Figure 3.

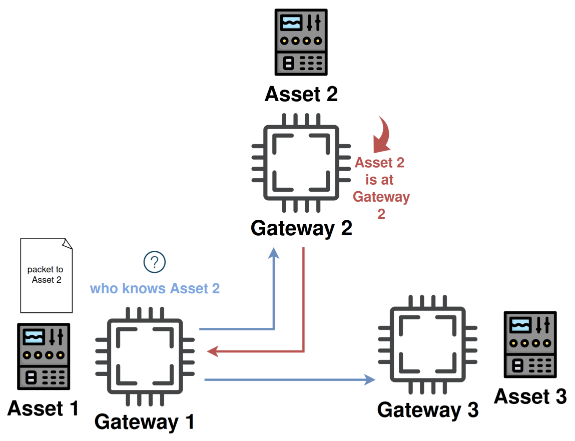


Figure 3: Visualization of endpoint gateway resolution in a multi-security gateway infrastructure

For a network of security gateways, a forward and reverse proxy implementation is already available as part of the KRITIS³M research project, which secures TCP traffic using TLS [31]. In our evaluation in Chapter VI, the proxy implementation serves as a benchmark for assessing the performance of the bump-in-the-wire security gateway. Based on the concept, a software design is developed and implemented which is described in the next chapter.

V. Implementation

In this chapter, the software design is further explained. The specific implementation and design details are provided in the following sections. The abstract concept of the application design is illustrated in Figure 4. The architecture features two independent interfaces: one for the asset and one for the tunnel. The protocols used within each interface should be as independent as possible from the gateway's logic, facilitating easy transitions to other protocols in future development. The software component L2 Gateway implements both the asset and tunnel interfaces but does not need to be aware of the concrete implementation. The design concept requires three methods for each interface: send, receive, and pipe. The send and receive methods are self-explanatory, while the pipe method handles the transition between interfaces, allowing for filtering and preprocessing before sending data. In future work, the pipe function could be used to implement the routing, to determine the target security gateway for a packet in a multi-security gateway infrastructure.

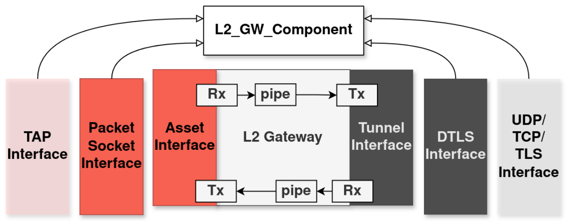


Figure 4: Abstract design of the application

Inspired by a blog post on dev.to, an extensible interface design was created, enabling functional polymorphism in C through a vtable with callback functions for concrete implementations of the `read`, `send`, and `pipe` functions [32]. For the VPN gateway application, concrete interfaces for UDP, TAP, packet socket, and DTLS are implemented, that can be used as tunnel and asset interfaces. This method allows the security gateway to be dynamically configured for a specific use case. For a bump-in-the-wire security gateway, the packet socket is used as the asset interface, and DTLS for the tunnel interface. Alternatively, the application can be deployed as a software service on an existing system, using the TAP interface as the asset interface. In cases where a VPN tunnel needs to be established without cryptographic security, UDP can be used for the tunnel.

The application is divided into two phases: initialization and operation. These phases are explained below. **Initialization phase**: In this phase, the application is set up and no packets from the asset can be transmitted. The DTLS client then attempts to establish a connection with the counterpart security gateway. Once both the client and server are connected, the VPN tunnel is established, the operation phase begins. **Operation phase**: As Illustrated in Figure 5, the operation phase handles data transfer asynchronously.

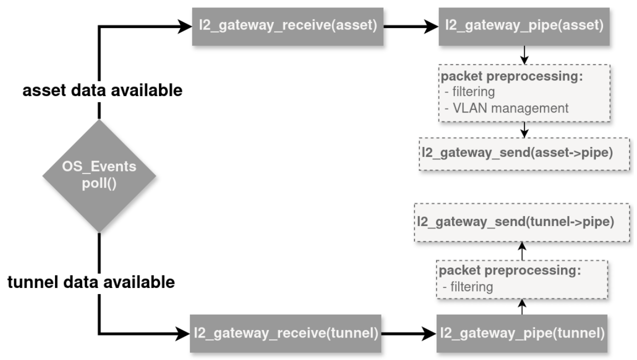


Figure 5: Visualization of the program flow in the operation phase.

During initialization, the DTLS client and server, as well as the packet socket, register for read, write, and error events using file descriptors. The logic is handled in a loop. When an event occurs, the `poll()` function returns. Then, the matching file descriptor for the detached event is determined. If data is available, the corresponding receive function is called, and the application flow proceeds as shown in Figure 5. This asynchronous approach efficiently manages multiple events concurrently, making it particularly useful for applications that require high performance or operate on platforms with limited resources. If an error occurs, the application closes existing connections and reverts back to the initialization phase. With the implementation in place, the subsequent chapter will pivot towards the evaluation of the application.

VI. Evaluation

This chapter evaluates the implementation by verifying the functionality and measuring the performance, and compares the results to an existing KRITIS3M proxy application. The performance is evaluated by measuring the Round Trip Time (RTT) and throughput. The results are discussed in Chapter VII. The test setup involved two security gateways using the VPN gateway application. Figure 7 visualizes the test setup. A Raspberry Pi 4 (running kernel version 6.1.21) and a Lenovo laptop equipped with 32 GB of RAM and an Intel Core i7-1260P processor, running Pop OS (kernel version 6.6.10), were utilized. Each device simulated both, the asset and the security gateway and are directly connected via a 1000BASE-T Ethernet cable.

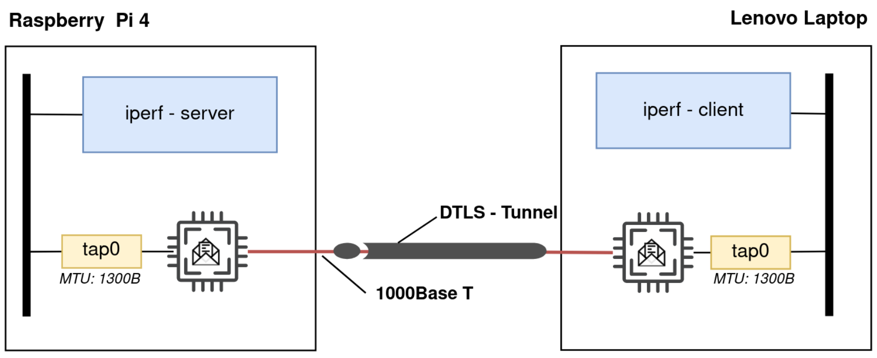


Figure 6: Test setup for the performance measurement: The Raspberry Pi 4 is configured as the iperf server, while the Lenovo laptop is set up as the iperf client.

Consequently, a TAP interface is used for the asset interface instead of the packet socket implementation. The applications for the measurements were built in release mode with optimization level 3 enabled. After the handshake phase, the data throughput is measured. The RTT, which measures the latency of the VPN, is assessed based on data derived from throughput measurements. The connection between the two security modules was established via a single physical connection without any other network participants, providing an ideal environment for the measurements. As a measurement tool, iperf3 was selected, which is a widely used tool for measuring network performance. The Raspberry Pi was configured as iperf server and the laptop as iperf client. For each objective, 100 measurements were taken, with data sent at 2-second intervals. The performance measurement of the proxy application was conducted in the same manner. The laptop hosts the iperf client and the forward proxy, which establishes a connection with the iperf server on the Raspberry Pi. The iperf server in turn is located behind the reverse proxy application. Figure 7 visualizes the throughput measurements of both applications.

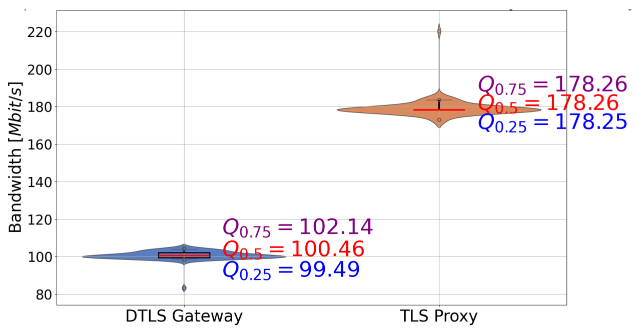


Figure 7: Throughput comparison between the VPN and proxy application

The mean throughput for the VPN application was 102 MBit/s. In comparison, the proxy application achieves 178 Mbit/s and therefore exceeds by a factor of almost 175 %. The distribution of both measurements is similar. Figure 8 visualizes the RTT measurements of both applications.

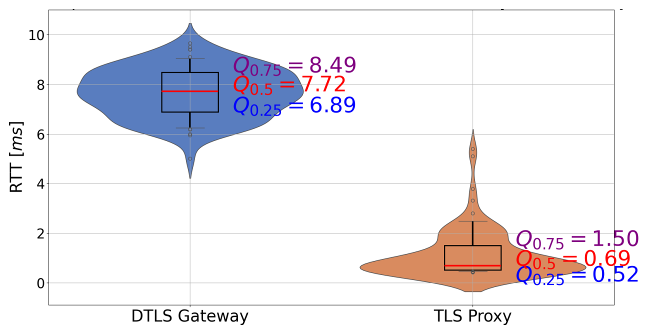


Figure 8: RTT comparison between the VPN and proxy application

The mean RTT for the VPN application was 7.68 ms, while the proxy application achieved 1.4 ms and is thus faster by a factor of five. These measurements confirm the performance and functionality of the implemented application. The following discussion compares and interprets the result and subsequently highlights the design limitations.

VII. Discussion

The primary objective of this research was to address the scientific question: **How can a bump-in-the-wire security gateway be implemented that operates on Layer 2?** To achieve this, Linux packet sockets and DTLS client and server implementations were utilized within the L2_Gateway module introduced in Chapter V. This module provides a flexible interface design, enabling the use of various protocols for the asset and tunnel interfaces, which is demonstrated in chapter Evaluation. The previous chapter validates the correct functionality of the VPN application and compares it to the existing proxy design.

With a mean throughput of 102 MBit/s, the VPN application is capable of handling typical network loads. However, the proxy application outperforms the VPN in both throughput and RTT measurements. The primary reason for the performance disparity is likely based on the increased data size that needs to be encrypted in case of the VPN tunnel. During the bandwidth measurement, data is transmitted in only one direction. In the case of the TLS proxy, only the data sent from the iperf client to the iperf server is encrypted on the TCP layer. However, with the VPN gateway, the TCP acknowledgement messages that are sent in response from the server to the client are also encrypted. This results in an increased load on the singe thread handling the DTLS connection, as these packets must be decrypted. As the test is CPU bound on the Raspberry Pi and only a single core is used, the achieved throughput is lower. In the future, the DTLS handling code could be extended to use two threads for sending and receiving, which should increase the throughput to be on roughly on par with the proxy application.

Another additional reason for this difference is the overhead introduced by the encapsulation mechanism discussed in Chapter IV. In comparison to the proxy, the tunnel's overhead is significantly greater, as the proxy only applies TLS to the packet and does not use encapsulation. As a negative side effect of the encapsulation, the MTU of the asset interface was reduced for the VPN application, which also results in a lower information density.

Despite these findings, there is still potential to improve the VPN's performance. Currently, the throughput is limited by the CPU, since the application runs on a single core. Theoretically, utilizing all four cores of the Raspberry Pi could increase performance by a factor of four. Additionally, WolfSSL performs cryptographic functions in software. Offloading these functions to hardware peripherals could alleviate CPU load. Therefore, the LaS³ is researching in a combination of software and hardware peripherals to provide a solution that meets the requirements.

VIII. Conclusion

The objective of this project was to implement a bump-in-the-wire security gateway. Therefore, a VPN application was created that utilizes a packet socket to capture packets from the asset at layer 2. These packets are subsequently encapsulated in DTLS and transmitted to the counterpart. The security gateway is capable of managing standard network traffic while remaining transparent in the communication chain, providing bump-in-the-wire security. Future research will include the management of security modules within the KRITIS IT infrastructure. Software-defined networking will be utilized to separate the network management (control plane) from the data transfer (data plane).

Acknowledgment

The presented work is part of the research project *KRITIS Scalable Safe and Secure Modules* (KRITIS³M), which is funded by the Project Management Jülich (PtJ) and the German Federal Ministry for Economic Affairs and Climate Action (BMWK) under funding code 03EI6089A.

References

[1] *Global cybercrime estimated cost 2028 — statista*. [Online]. Available: https://www.statista.com/forecasts/1280009/cost-cybercrime-worldwide (visited on 03/08/2024).

[2] P. I. LLC, "Ninth anual cost of cybercrime: UNLOCKING THE VALUE OF IMPROVED CYBERSECURITY PROTECTION," *Accenture security*, 2019.

[3] O. Schily, "Es gibt keine sicherheit ohne IT-sicherheit. otto schily anlässlich der fachkonferenz des „münchener kreises". münchen.," Münchener Kreises, 2003.

[4] Microsoft, *Microsoft digital defense report*, Oct. 2021. (visited on 04/18/2024).

[5] S. d. Wissenschaft, *SolarWinds: Ein hackerangriff, der um die welt geht*, Jan. 2021. [Online]. Available: https://www.spektrum.de/news/solarwinds-ein-hackerangriff-der-um-die-welt-geht/1819187 (visited on 04/18/2024).

[6] M. Siegfried and N. Pohlmann, "Warum auf lange sicht die IT die OT managen muss," *IT-Sicherheit*, 2022.

[7] *BSI - KRITIS verordnung*. [Online]. Available: https://www.bsi.bund.de/DE/Themen/KRITIS-und-regulierte-Unternehmen/Kritische-Infrastrukturen/Allgemeine-Infos-zu-KRITIS/Stand-der-Technik-umsetzen/stand-der-technik-umsetzen_node.html (visited on 03/08/2024).

[8] J. Birkmann, C. Bach, S. Guhl, M. Witting, T. Welle, and M. Schmude, "State of the art der forschung zur verwundbarkeit kritischer infrastrukturen am beispiel strom/stromausfall,"

[9] *Krypto-agilität - glossar - prof. dr. norbert pohlmann*. [Online]. Available: https://norbert-pohlmann.com/glossar-cyber-sicherheit/krypto-agilitaet/ (visited on 05/20/2024).

[10] J. Staggs and S. Shenoi, Eds., *Critical Infrastructure Protection XVI: 16th IFIP WG 11.10 International Conference, ICCIP 2022, Virtual Event, March 14–15, 2022, Revised Selected Papers*, vol. 666, IFIP Advances in Information and Communication Technology, Cham: Springer Nature Switzerland, 2022, ISBN: 978-3-031-20136-3 978-3-031-20137-0. DOI: 10.1007/978-3-031-20137-0. [Online]. Available: https://link.springer.com/10.1007/978-3-031-20137-0 (visited on 06/01/2024).

[11] R. J. H. Jr. "The importance of operational technology (OT) systems to maintain a secure standard of living in today's modern society," Fortinet Blog. Section: Industry Trends. (Dec. 6, 2017), [Online]. Available: https://www.fortinet.com/blog/business-and-technology/the-importance-of-operational-technology-ot-systems-for-a-standard-of-living-in-today-s-modern-society (visited on 05/27/2024).

[12] National Institute of Standards and Technology, "Framework for improving critical infrastructure cybersecurity, version 1.1," National Institute of Standards and Technology, Gaithersburg, MD, NIST CSWP 04162018, Apr. 16, 2018, NIST CSWP 04162018. DOI: 10.6028/NIST.CSWP.04162018. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf (visited on 06/05/2024).

[13] D. Meng, "Implementation of a host-to-host VPN based on UDP tunnel and OpenVPN tap interface in java and its performance analysis," Shandong University, 2013. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6554047 (visited on 03/01/2024).

[14] *What is TCP meltdown? — OpenVPN*. [Online]. Available: https://openvpn.net/faq/what-is-tcp-meltdown/ (visited on 04/10/2024).

[15] O. Honda, H. Ohsaki, M. Imase, M. Ishizuka, and J. Murayama, “Understanding TCP over TCP: Effects of TCP tunneling on end-to-end throughput and latency,” *Proc SPIE*, 2005.

[16] S. Gallenmüller, D. Schöffmann, D. Scholz, F. Geyer, and G. Carle, “DTLS performance-how expensive is security?,” 2019.

[17] B. für Sicherheit in der Informationstechnik, “VS – Anforderungsprofil VPN-Gateway,” Mar. 12, 2021, p. 29. [Online]. Available: https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Zulassung/VS-Anforderungsprofile/BSI-VS-AP-0001.pdf?__blob=publicationFile&v=2 (visited on 05/21/2024).

[18] S. T. Aung and T. Thein, “Comparative analysis of site-to-site layer 2 virtual private networks,” in *2020 IEEE Conference on Computer Applications(ICCA)*, Yangon, Myanmar: IEEE, Feb. 2020, pp. 1–5, ISBN: 978-1-72815-925-6. DOI: 10.1109/ICCA49400.2020.9022848. [Online]. Available: https://ieeexplore.ieee.org/document/9022848/ (visited on 06/05/2024).

[19] *Packet(7) - linux manual page*. [Online]. Available: https://man7.org/linux/man-pages/man7/packet.7.html.

[20] *Promiskuitiver modus – wikipedia*. [Online]. Available: https://de.wikipedia.org/wiki/Promiskuitiver_Modus (visited on 05/07/2024).

[21] “Virtual networking 101: Bridging the gap to understanding TAP,” The Cloudflare Blog. (Oct. 6, 2023), [Online]. Available: https://blog.cloudflare.com/virtual-networking-101-understanding-tap (visited on 06/05/2024).

[22] N. Modadugu and E. Rescorla, “The design and implementation of datagram TLS,” *Network and Distributed System Security (NDSS) Symposium*, Dec. 2003.

[23] *RFC 9147 - the datagram transport layer security (DTLS) protocol version 1.3*, Publication Title: Internet Engineering Task Force (IETF), Apr. 2022. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc9147 (visited on 05/06/2024).

[24] B. f. S. i. d. Informationstechnik, *BSI - public key infrastrukturen - public key infrastrukturen (PKIen)*. [Online]. Available: https://www.bsi.bund.de/DE/Themen/Oeffentliche-Verwaltung/Elektronische-Identitaeten/Public-Key-Infrastrukturen/public-key-infrastrukturen.html.

[25] Cloudflare, *Funktionsweise von mTLS*. [Online]. Available: https://www.cloudflare.com/de-de/learning/access-management/what-is-mutual-tls/ (visited on 05/13/2024).

[26] I. E. T. Force (IETF), *RFC 5246 - the transport layer security (TLS) protocol version 1.2*, Oct. 2008. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc5246#section-7.4 (visited on 04/27/2024).

[27] Cloudflare, *Zero trust security — what is a zero trust network?* [Online]. Available: https://www.cloudflare.com/learning/security/glossary/what-is-zero-trust/ (visited on 05/18/2024).

[28] “MOBILE ACCESS CAPABILITY PACKAGE v2.1,” *National Security Agency (NSA)*, 2018.

[29] *IEEE Standard for Management Information Base (MIB) Definitions for Ethernet*, ISBN: 9780738184326. DOI: 10.1109/IEEESTD.2013.6572796. [Online]. Available: http://ieeexplore.ieee.org/document/6572796/ (visited on 06/07/2024).

[30] *Behebung von IPv4-Fragmentierung, MTU-, MSS- und PMTUD-Problemen mit GRE und IPSec*, de. [Online]. Available: https://www.cisco.com/c/de_de/support/docs/ip/generic-routing-encapsulation-gre/25885-pmtud-ipfrag.html (visited on 06/07/2024).

[31] T. Frauenschläger and J. Mottok, “Problems and New Approaches for Crypto-Agility in Operational Technology,” in *12th European Congress on Embedded Real Time Software and Systems (ERTS 2024)*, Toulouse, France, Jun. 2024. [Online]. Available: https://hal.science/hal-04614197.

[32] Github: TotallyNotChase, *Typeclass-interface-pattern: Interface based polymorphism pattern for standard c*. [Online]. Available: https://github.com/TotallyNotChase/typeclass-interface-pattern (visited on 03/02/2024).

Effect of Compiler Optimization Flags on SipHash algorithm Side-Channel Information Leakage

Matúš Olekšák, Vojtěch Miškovský
Department of Digital Design
Faculty of Information Technology
CTU in Prague, Czech Republic
{oleksmat,miskovoj}@fit.cvut.cz

***Abstract*—This work presents an experimental evaluation of influence of compiler optimization flags on side-channel information leakage. SipHash was used as a reference algorithm an ARX-based pseudorandom function optimized for short inputs. ChipWhisperer CW308 with various targets was used for the evaluation using guessing entropy of CPA and Welch's t-test. The main contribution of this paper is analysis of impact of each flag and its suitability for implementations minimizing side-channel leakage. This work was submitted to DSD 2024.**

***Index Terms*—compiler, GCC, SipHash, ARX, side-channel, ChipWhisperer**

I. Introduction

Side-channel attacks are one of the most serious threats to the security of embedded devices as they can break cryptographically secure algorithms. These attacks exploit the fact that the physical properties of the operating cryptography device depend on the data being processed. Side channels include power consumption [?], electromagnetic radiation [?], and even sound [?]. This paper focuses on power consumption.

The leakage of sensitive information in the side channel is influenced by the processor, architecture, machine code and the environment in which the algorithm is executed. However, compiler optimizations has a significant impact on the output of the machine code, resulting in lateral channel leakage. To find out how much difference it makes, tests should be carried out on several architectures commonly used for embedded development.

SipHash [?] was used as the reference algorithm for this paper. It is an ARX-based pseudorandom function, and there was a successful CPA attack on it [?], which was used for comparison between different optimizations in this paper. This algorithm became of interest as it is used in modern automobile platforms.

II. Background

Optimization flag [?] is compiler parameter, which influence performance and/or code size at the expense of compilation time and possibly the ability to debug the program. The difference between the used instructions and the order of the binary code affects the leakage of the side channel. The main contribution of this work is the evaluation of how different optimization flags affect the leakage of the side channel.

There are optimization flags designed for debugging purposes, such as -Og or -O0, but they are not used in production, therefore they are omitted in this work. Some are used to reduce binary size (-Os and -Oz) and most embedded applications are assumed to use -Os. The remaining optimization flags (-O1, -O2, -O3 and -Ofast) are used to improve performance. However, -Ofast disregards strict standards compliance, and is therefore not intended to be used in production, but it was measured for its completeness.

Optimized binary code affects many aspects, e.g., better usage of registers and reduced data transport over the bus. However, it is not the case, the more optimized binary, the less it leaks sensitive information. But there has been no work to consider compiler optimization flags as a source of side-channel leakage or as a countermeasure. As a result, it was decided to measure how optimizations affect the side-channel leakage.

A. SipHash

SipHash is an ARX-based pseudorandom function, which can be used to generate 64-bit MACs. There are four state variables called $v0$-$v3$, the default value of which is "somepseudorandomly generatedbytes" in ASCII. However, other initial values can be used as long as $v0$ and $v1$ are different from $v2$ and $v3$. To make it clear, in this text, the subscript in the name of the variable represents the round number and represents the value of the variable at the beginning of the selected round, variable without subscript represents the initial value.

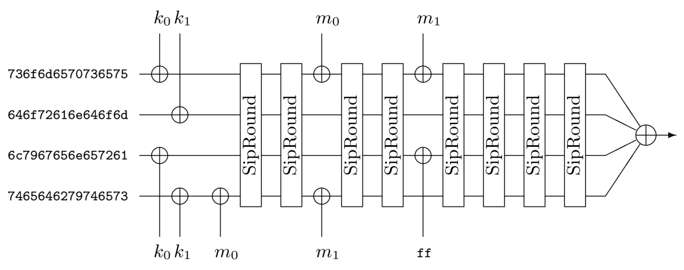


Fig. 1: SipHash Diagram [?].

The 128-bit key is half-transformed to two 64-bit values of $k0$ and $k1$. Before the first round of transformation, $v0_1 = v0 \oplus k0$, $v1_1 = v1 \oplus k1$, $v2_1 = v2 \oplus k0$, $v3_1 = v3 \oplus k1 \oplus m0$, $m0$ represents the first 64-bit text. The default number of rounds is two per 64-bit plain text and four final rounds.

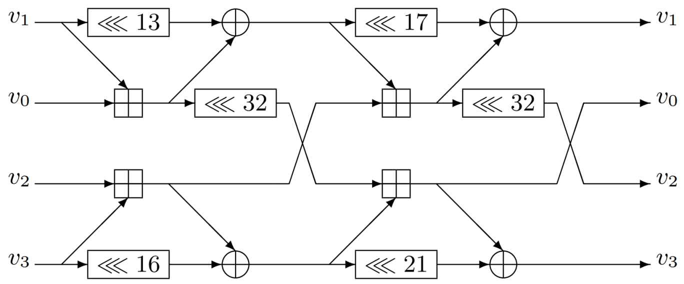


Fig. 2: SipHash Round Diagram [**?**].

B. CPA Attack

In 2004, Brier et al. [**?**] introduced the correlation power analysis (CPA), which is a non-profiled side-channel attack. At the beginning of the attack, the power consumption is measured. After this physical prerequisite, all other steps are purely computational. The leakage function must be selected according to the attacked algorithm and its implementation. Secret information (usually the encryption key) must be divided into small parts (e.g. bytes) in order to make its enumeration calculations feasible (e.g. 256 possible key candidates for one byte). Then, using the chosen leakage function, the power consumption of each key candidate is estimated. In the final step, the correlation between the measured and estimated power consumption is calculated. The most correlating key candidate is considered the correct key.

There is a CPA attack [**?**] on SipHash, which is used as a base for lightened attack used in this paper. In comparison with the originally proposed attack, only the first part of the attack was used in order to retrieve the fifth byte of $k1$ in this work. This approach is sufficient because it only visualizes the difference between leakage of the side channel of different optimizations. More information on the attack consists of the intermediate value $f_{k1}(m0) = v3 \oplus k1 \oplus m0$. As a leakage function, $-HW(f_{k1}(m0))$ is used where $HW(x)$ is Hamming Weight of x. The Pearson correlation between the estimated power consumption by leakage function and the measured power traces is used to distinguish the best key candidate.

III. Related work

To the best of our knowledge, there is no publication on compiler optimization flags and side-channel leakage measurement. The most related publications deal with the security vulnerabilities that are generated by a compiler [**?**] and the detection of specific instructions. There is research paper on the estimation of electromagnetic radiation by the execution of a particular sequence of instructions [**?**]. Another publication [**?**] refers to methods for code generators (not compilers), which preserve first-order security by taking into account specific CPU leakage characteristics. And there is also a publication, where the authors have developed a method of profiling [**?**], which is capable of detecting executed instructions.

IV. Experiment setup

Initially, the power traces were measured for all the optimizations (-Oz, -Os, -O1, -O2, -O3 and -Ofast). The ChipWhisperer CW308 UFO Board was used to measure three different targets: STM32F0, XMEGA, and MPC5676R to cover the most commonly used architectures for embedded development. Reference implementation [**?**] of SipHash algorithm was ported to ChipWhisperer and used afterwards. A CPA attack was carried out and the guessing entropy [**?**] was calculated in range from 2 to 2000 power traces. Values of guessing entropy in tables are calculated from 2 000 power traces. For better visualization of differences, graphs of correlation coefficients for key assumptions were generated. Blue line visualizes correct key correlation coefficient, the yellow line is for $plaintext \oplus v3$ and the red line for all other key candidates. First testing measurements were using 20 000 power traces but it was verified, that even only 2 000 power traces shows the same correlation peaks as higher number of traces, and therefore all shown measurements are generated from 2 000 power traces.

We also performed non-specific Welch's t-test [**?**] using fixed vs. random plain text and fixed key. It is used to test the hypothesis that two groups have equal means, which can be used to test whether device behave differently with fixed and random plaintext or cipher key.

A. STM32F0

The ChipWhisperer target used for the STM32 platform is the CW308T-STM32F. More precisely it is the STM32F0, which contains a STM32F071RBT6 processor with ARM Cortex M0 core and 128 KB of flash.

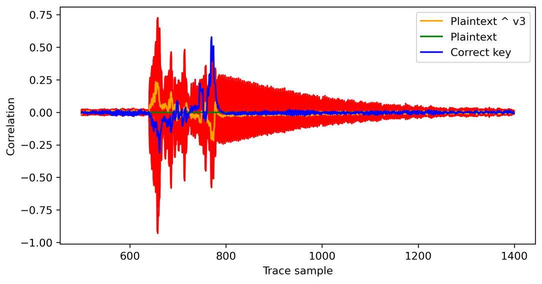
Binary files were generated using arm-none-eabi-gcc 13.2.0. The -Oz and -Os optimization flags produce the same binary file, therefore -Oz is omitted. A summary of binary size and guessing entropy for 2000 power traces is in Table I.

Correlation coefficient graphs of STM32F0 are shown on Figure 3. Optimizations -Os (a) and -O1 (b) show significant correlation peak, in contrary to -O2 (c), which has no point with the highest correlation for the best key candidate. -O3 (d) has a single peak, but it is approximately the same size as other 3 peaks. The last one -Ofast (e) looks almost exactly the same like -O3.

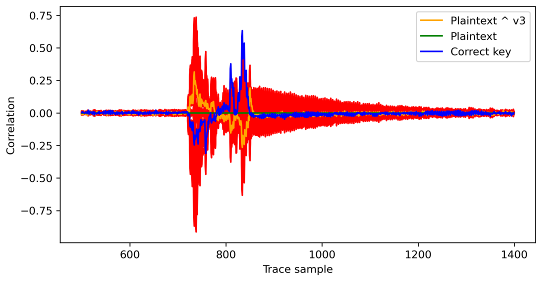
TABLE I: A table summarizing the binary size of the different optimization flags of the STM32F0 device.

Optimization flag	Binary size	Guessing Entropy
-Os	12 564 B	10
-O1	13 596 B	9
-O2	13 592 B	159
-O3	15 708 B	22
-Ofast	15 480 B	32

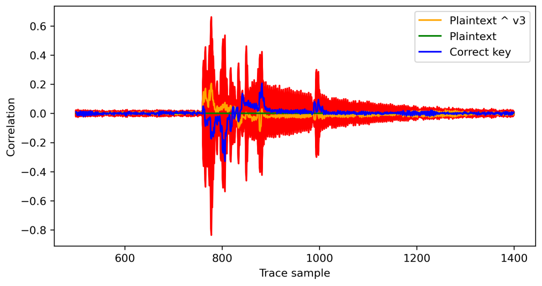
Guessing entropy for all of the optimizations is shown on Figure 4. It is visible that guessing entropy is stabilized from approximately 300 power traces. The flags -O1 and -Os have the lowest guessing entropy. -O3 and -Ofast have slightly higher guessing entropy, in range from 20 to 40. The -O2 flag has made the implementation to have even higher guessing entropy value than 150.



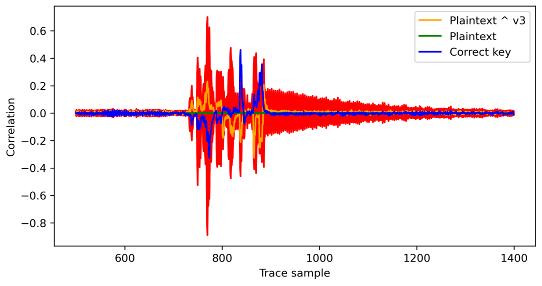
(a) -Os



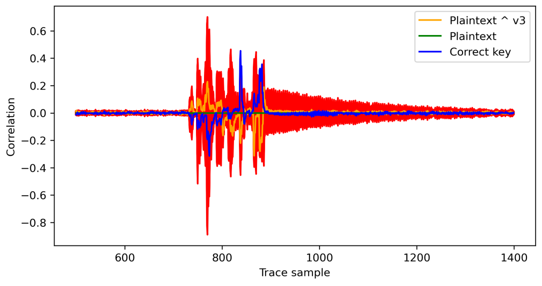
(b) -O1



(c) -O2



(d) -O3



(e) -Ofast

Fig. 3: Graphs of correlation coefficients for STM32 device.

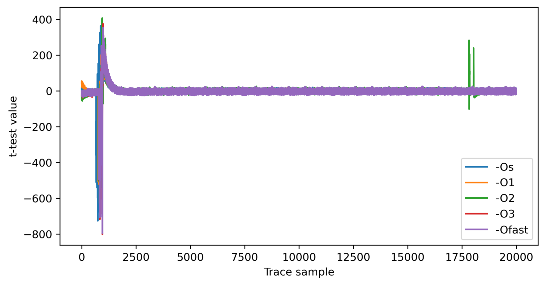


Fig. 5: Graph of Welch t-test for STM32F0 device.

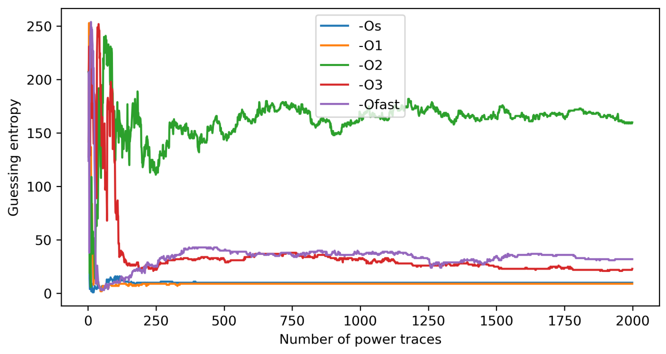


Fig. 4: Graph of STM32F0 Guessing entropy.

B. XMEGA

The CW308T-XMEGA target was used for the XMEGA platform. It is based on ATXmega128D4-AU processor with the AVR core architecture and 128 KB of flash.

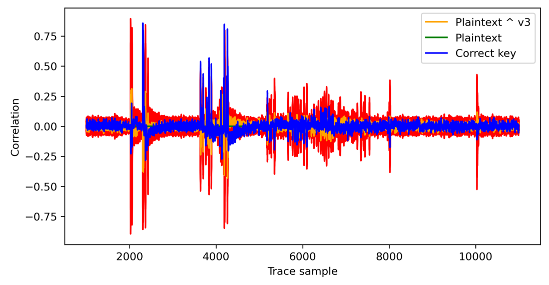
Compiler avr-gcc 13.2.0 was used for this platform. The -Oz and -Os optimization flags generate the same binary file, therefore -Oz is omitted. A summary of binary size and guessing entropy for 2000 power traces is in Table II.

Correlation coefficient graphs of XMEGA are shown on Figure 6. All graphs (a, c, d, e) except -O1 (b) have correlation peaks of correct key, therefore they are vulnerable to the originally presented attack. Even -O1 with some postprocessing can be successfully attacked.

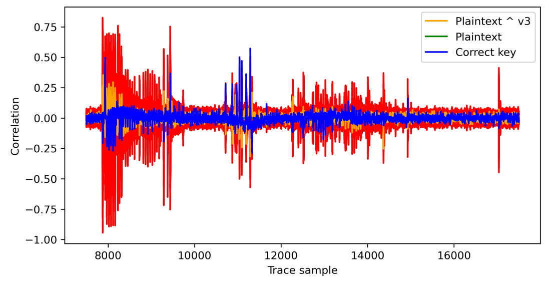
TABLE II: A table summarizing the binary size of the different optimization flags of the XMEGA device.

Optimization flag	Binary size	Guessing Entropy
-Os	6 912 B	2
-O1	8 296 B	22
-O2	6 622 B	1
-O3	10 674 B	1
-Ofast	10 674 B	1

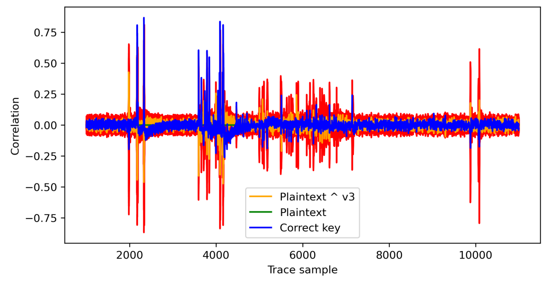
Due to the correlation peaks occurring in all optimization flags with the exception of -O1, the guessing entropy graph on Figure 7 is not very diverse. -O1 optimization flag has guessing entropy around 20, while all the other flags have value of approximately 1.



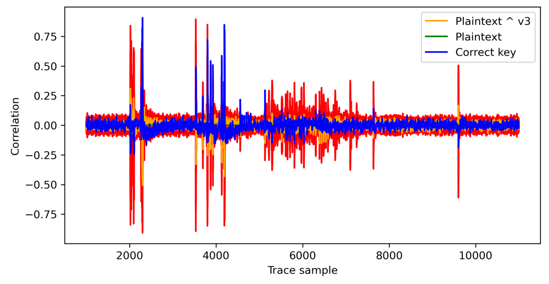
(a) -Os



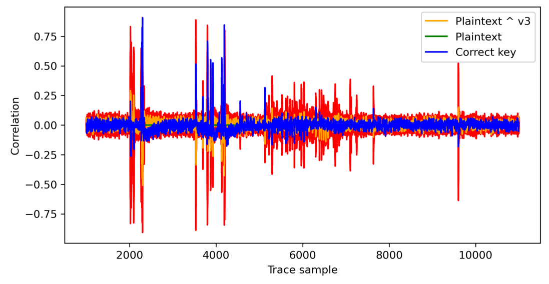
(b) -O1



(c) -O2



(d) -O3



(e) -Ofast

Fig. 6: Graphs of correlation coefficients for XMEGA device.

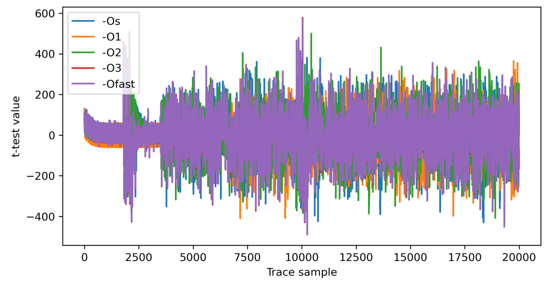


Fig. 8: Graph of Welch t-test for XMEGA device.

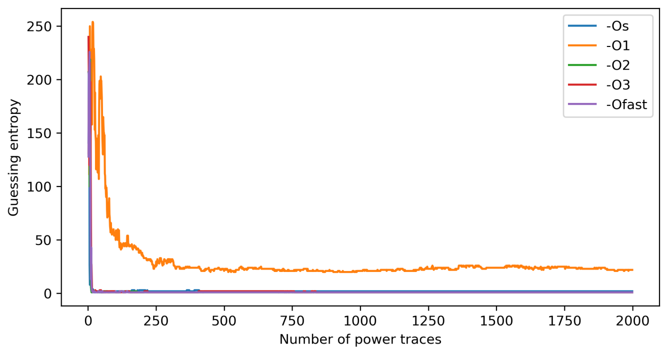


Fig. 7: Graph of XMEGA Guessing entropy.

C. *MPC5676R*

The PowerPC architecture is represented by the ChipWhisperer target CW308T-MPC5676R. It contains a SPC5676RDK3MVU1R processor with an e200z7 core architecture and 6 MB of flash.

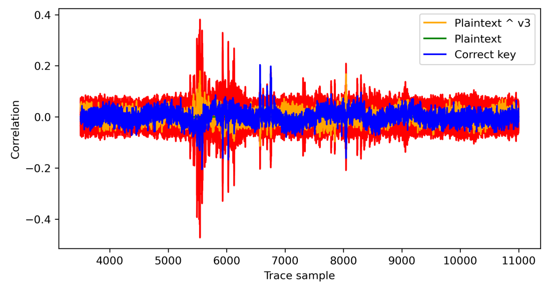
The PowerPC architecture uses the powerpc-eabivle-gcc 4.9.4 compiler, which is embedded in the S32 Design Studio for Power Architecture v2.1. A summary of binary size and guessing entropy for 2000 power traces is in Table III.

Correlation coefficient graphs of MPC5676R are shown on Figure 9. Graphs of -Os (a) and -O1 (b) optimization contain peaks of correlation of correct key, but it is not the highest point. Rest of optimization flags (c, d, e) has no peaks, and therefore it is not vulnerable to that specific attack.

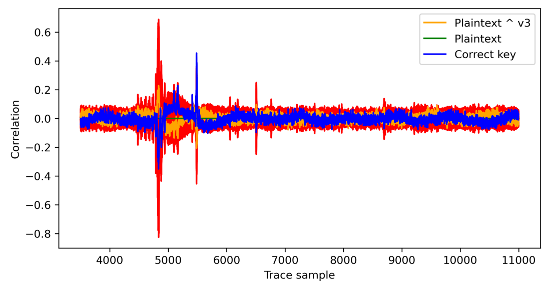
TABLE III: A table summarizing the binary size of the different optimization flags of the MPC5676R device.

Optimization flag	Binary size	Guessing Entropy
-Os	13 550 B	51
-O1	14 066 B	16
-O2	13 938 B	200
-O3	15 730 B	167
-Ofast	15 730 B	106

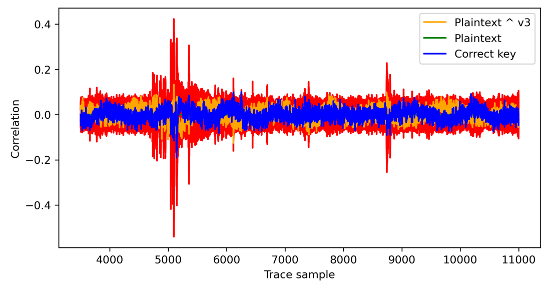
The guessing entropy of all optimizations is shown on Figure 10. For the first 300 power traces the guessing entropy oscillates through the entire spectrum of values. From 1300 power traces it is stabilized. -O1 and -Os are the least secure



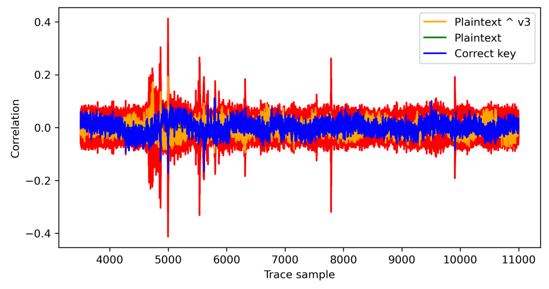
(a) -Os



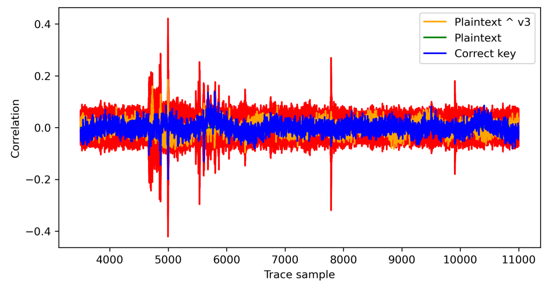
(b) -O1



(c) -O2



(d) -O3



(e) -Ofast

Fig. 9: Graphs of correlation coefficients for MPC5676R device.

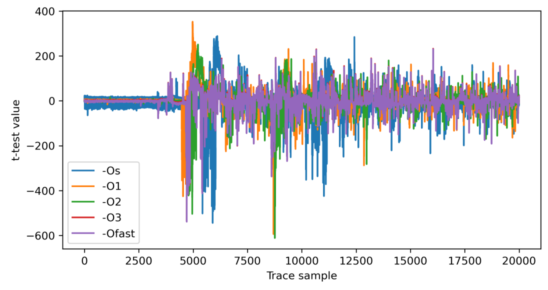


Fig. 11: Graph of Welch t-test for MPC5676R device.

flags, while the other flags have a guessing entropy value of more than 100.

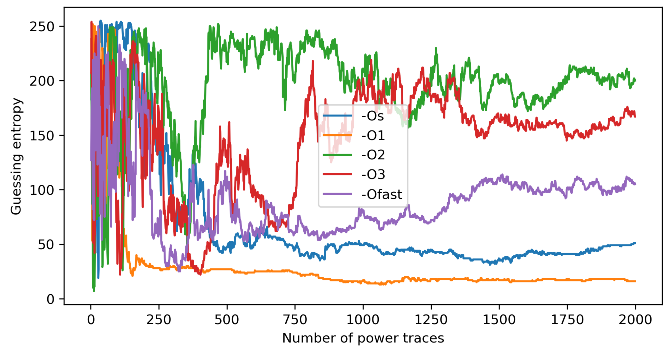


Fig. 10: Graph of MPC5676R Guessing entropy.

V. Flag evaluation

To find out which flags caused difference between -O1 and -O2, the -O2 flag was replaced by all optimization flags it includes and sequentially it was measured with all but one.

The difference of guessing entropy between -O1 and -O2 for all tested architectures was caused by flags -fexpensive-optimizations, -falign-labels, -fcrossjumping, -finline-small-functions, -freorder-blocks-algorithm=stc, -fgcse, -fschedule-insns2, -ftree-loop-vectorize, -ftree-vrp, -fstrict-aliasing. The reduction in STM32F0's guessing entropy of the -O3 optimization flag was caused by -fpeel-loops, -funswitch-loops and -fversion-loops-for-strides.

However, there is no noticeable difference in t-value amplitude between the different optimization flags, only slightly different graph courses caused by different binary codes visible in Figure 8 and 11. The only exception is the -O2 optimization flag on the STM32F0 platform, it can be seen in Figure 5, where the t-value peak is visible around sample 18000.

VI. Conclusion

The measurements clearly showed that there is a significant difference between the optimization flags in terms of side-channel leakage. That implies developers of embedded systems using not only SipHash but also other cryptographic algorithms must be aware of this phenomenon. Particularly for SipHash on the STM32F0 platform, the flags that make implementation more vulnerable are -fpeel-loops, -funswitch-loops and -fversion-loops-for-strides. It is most evident in the guessing entropy. However, the results of Welch's t-test did not distinguish the various optimizations. This implies that there is a difference in the distribution of the leakage rather than in its quantity. Therefore, the side-channel attacks, including the one which is used in this case study, must adapt to different optimization flags. This should be the subject of further research.

ACKNOWLEDGMENT

This work has been partially supported by the CTU project SGS23/208/OHK3/3T/18.

Vulnerability Assessment of OPC UA Server Implementations

Sacha Colucci and Jürgen Mottok
Technical University of Applied Sciences Regensburg
Faculty of Electrical Engineering and Information Technology
Seybothstraße 2, Regensburg, 93053, Germany
{sacha.colucci, juergen.mottok}@oth-regensburg.de

Abstract

Critical infrastructures are essential to daily life, whether for individuals at home or companies producing goods, because humanity depends on them. Therefore, the communication of the components controlling these infrastructures must be protected from cyberattacks. The most commonly used protocol in such critical infrastructures is the OPC UA protocol, also known as the IEC-62541 standard. This research looks into security gaps within OPC UA server implementation by performing a vulnerability assessment. Furthermore, this research looks into the generation of network traffic data for intrusion detection and prevention systems (IDS/IPS). The results revealed vulnerabilities across all tested implementations and emphasize the critical need for prioritizing security throughout the development lifecycle of OPC servers. Additionally, the generated traffic data presents a unique opportunity to develop and evaluate machine learning-based IDS or IPS solutions specifically tailored to OPC UA security.

***Keywords*— OPC UA, IEC-62541, vulnerability assessment, network traffic dataset, IDS/IPS, cybersecurity**

I. Introduction

As pointed out by the Check Point Research Team [1], cyberattacks are becoming more and more frequent. The research team reports a 38% increase in cyberattacks globally between 2021 and 2022, with the most frequently attacked industries being education, research, government, and healthcare. For this reason, the German government enacted the IT Security Act 2.0 [2], which sets minimum requirements for security measures against cyberattacks. In this context, the KRITIS³M research project [3, 4] was launched to develop a suitable and cost-effective solution for critical infrastructures, with a focus on energy and water infrastructures. The goal of this project is to develop a product that ensures a consistently high level of security over the entire lifetime of the components used, which can be up to 30 years. In addition, the development should allow for retrofitting so that the solution can not only be integrated into newly installed devices, but existing devices can also be upgraded with it. To achieve this, both the hardware and the software must always be adapted in a timely manner to the latest regulations of the institutions such as BSI [5], ENISA [6] or NIST [7].

A. Problem Area

One of the key components of critical infrastructures is the IEC-62541 standard, also known as the Open Platform Communications Unified Architecture (OPC UA) protocol, which is widely used in industrial automation. OPC UA is a communication protocol that enables secure and reliable data exchange between industrial devices and systems. It is used in various industries, including manufacturing, energy, and transportation, to enable interoperability between different devices and systems. Given the critical nature of these industries, security is of utmost importance when implementing OPC UA systems. However, like any other software system, OPC UA implementations are susceptible to vulnerabilities, as shown by the Security Bulletin of the OPC Foundation [8]. On this bulletin, the OPC Foundation lists known vulnerabilities in the OPC UA protocol and provides a risk assessment for each vulnerability. These vulnerabilities can pose serious risks to the security and integrity of industrial systems, because they can be exploited by attackers to gain unauthorized access, disrupt operations, or steal sensitive information. Therefore, it is essential to conduct vulnerability assessments of OPC UA server implementations to identify and mitigate potential security risks.

B. Motivation

This research aims to contribute to the field of OPC UA security in multifaceted ways. Primarily, it focuses on conducting a vulnerability assessment of multiple OPC UA server implementations. By identifying and analyzing these vulnerabilities, the research seeks to improve the overall security posture of OPC UA systems and mitigate potential risks. These findings can inform developers, system administrators, and security professionals about common attack vectors and equip them with knowledge to detect malicious traffic. Furthermore, this research extends beyond vulnerability assessment to generate valuable data for intrusion detection and prevention systems (IDS/IPS). The captured traffic data, encompassing both normal and malicious behavior, serves as a critical training ground for machine learning algorithms. By training on this comprehensive dataset, machine learning models can be fine-tuned to effectively identify and thwart future cyberattacks targeting OPC UA systems. Additionally, the assessment process provides valuable insights into the performance characteristics of various OPC UA server implementations. By analyzing factors like response times, resource utilization, and scalability under stress conditions, this research can contribute to performance profiling efforts. This information can be crucial for system administrators when selecting and configuring OPC UA servers for optimal performance in real-world deployments. Ultimately, this research has the potential to significantly enhance the security and overall effectiveness of OPC UA systems in critical infrastructure environments. By uncovering vulnerabilities, informing security practices, and aiding in the development of robust IDS and IPS solutions, this research can help safeguard these vital systems from cyber threats.

C. Outline

The remaining sections of this paper are structured as follows. Background information about vulnerability assessment and on the OPC UA protocol is provided in Section II. Then, related work in the field of vulnerability assessment, penetration testing, and OPC UA security analysis is discussed in Section III. The methodology used in this research for vulnerability assessment is described in Section IV. The implementation details of the vulnerability assessment, including the setup, server implementations, and tools used, are presented in Section V. The results of the vulnerability assessment, including the identified vulnerabilities, are presented in Section VI. The implications and reasons for the results are discussed in Section VII. Finally, the paper is summarized, and potential future work is suggested in Section VIII.

II. Background

This section provides the technical background knowledge required to understand the subsequent vulnerability assessment of OPC UA systems. This section firstly explains the definition and meaning of vulnerability assessment. The history and architecture of the OPC UA protocol is then briefly explained.

A. Vulnerability assessment

Since industrial automation systems are becoming larger and more complex, IT and OT networks are becoming more and more interconnected, allowing for improved control and monitoring of the systems. However, this integration also exposes them to a wider range of cybersecurity threats. Security in industrial automation is crucial to ensuring operational continuity, maintaining data integrity and confidentiality, and complying with regulatory requirements [9]. Furthermore, disruptions can lead to significant financial losses, safety hazards, and environmental damage [9, 10]. Therefore, vulnerability assessment is a critical component of a comprehensive cybersecurity strategy. Vulnerability assessment involves systematically identifying, classifying, and evaluating security flaws in a system [9]. This process typically utilizes automated tools to scan for known vulnerabilities, such as outdated software versions, misconfigurations, and weak passwords. The goal is to provide a comprehensive list of vulnerabilities that could potentially be exploited, along with an assessment of their severity. This allows organizations to prioritize remediation efforts based on the risk each vulnerability poses.

B. OPC Unified Architecture

Open Platform Communications Unified Architecture (OPC UA or IEC-62541) is a communication protocol that enables secure and reliable data exchange between industrial devices and systems. It emerged as a response to the shortcomings of its predecessors, OPC Data Access (OPC DA), OPC Alarms & Events (OPC AE), and OPC Historical Data Access (OPC HDA), together called "OPC Classic", which dominated industrial automation communication in the mid-1990s [11, 12]. While OPC Classic facilitated communication between devices, it lacked security features, couldn't function across different platforms, and struggled to keep pace with the growing demands of automation. Recognizing these limitations, the OPC Foundation embarked on developing OPC UA in 2003. The first version of OPC UA was released in 2006, marking a significant improvement over OPC Classic. The goal of this new standard was to provide a cross-platform, open-source standard for data exchange between various industrial devices, sensors, and even cloud applications, which also supports security functions such as encryption and authentication [11, 13]. A key benefit is its ability to promote interoperability, ensuring seamless communication between devices from different manufacturers. Additionally, it boasts a flexible design that allows for future development of new features without affecting existing systems. The OPC Foundation continues to actively support OPC UA, with recent efforts focused on integrating it with Artificial Intelligence and cloud technologies. The latest version of the OPC UA specification is version 1.05, which was released in 2023 [12].

The protocol is based on a layered architecture as shown in Figure 1, often called a service-oriented architecture (SOA), where communication is broken down into distinct layers, each with a specific function. The foundation is the OPC UA Services layer (red and blue block in Figure 1), where core functionalities like data reading, writing, and subscriptions are defined. Building on top of the foundation is the Information Model layer [14], which consists of the *Core Information Model* (yellow block in Figure 1), the *Companion Information Model* (green block in Figure 1) and vendor specific extensions. For more information about the OPC UA Services and Information Model layers, refer to the OPC UA Specifications [15, 16].

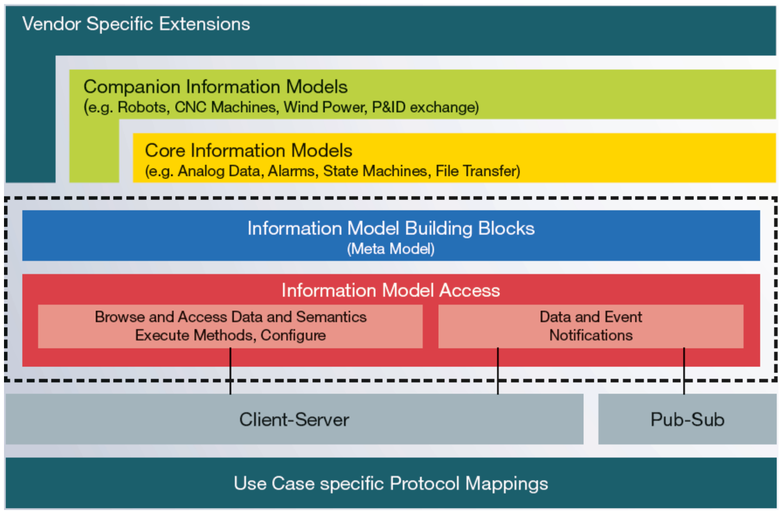

Vendor Specific Extensions
Companion Information Models
(e.g. Robots, CNC Machines, Wind Power, P&ID exchange)
Core Information Models
(e.g. Analog Data, Alarms, State Machines, File Transfer)
Information Model Building Blocks
(Meta Model)
Information Model Access
Browse and Access Data and Semantics
Execute Methods, Configure
Data and Event
Notifications
Client-Server
Pub-Sub
Use Case specific Protocol Mappings

Fig. 1. Architecture of the OPC UA protocol. [13]

III. Related Work

Several research efforts have been made to develop tools and frameworks to facilitate vulnerability assessment and penetration testing and boost their efficiency and effectiveness. Furthermore, various studies have been conducted to identify vulnerabilities in the OPC UA protocol and assess their impact on the security of industrial systems. This section provides an overview of the related work in the field of vulnerability assessment, penetration testing, and OPC UA security research.

The goal of Alhamed and Rahman [10] is to raise awareness of the importance of vulnerability assessment and penetration testing in networks. By conducting a systematic literature review of almost 40 papers and research publications, they identify commonly used tools and frameworks for vulnerability assessment and penetration testing, as well as the most common vulnerabilities and attack vectors in networked systems.

Bartusiak *et al.* [9] focus on automating security assessments, particularly for critical infrastructure. They introduce a novel methodology for conducting these assessments. This methodology helps analyze the gap between implemented security measures and the required security level, considering both user-specific requirements and recommendations from legal acts or standards. Furthermore, they present a possible automation strategy to streamline the process of conducting cybersecurity assessments.

The paper published by Sarker *et al.* [17] is a review article about penetration testing frameworks, standards, tools, and scoring methods. The authors provide a systematic literature review and comparison of the mentioned items to develop guidelines to facilitate the selection of the most suitable tools and methodologies. Sarker *et al.* also presents vulnerabilities that arise when performing penetration tests and their importance in the context of cybersecurity.

Ivanova *et al.* [18] investigate cybersecurity in OPC UA systems. In their paper, the authors identify attack scenarios and vulnerabilities that can be used to exploit OPC UA applications and systems. They also propose mitigation strategies to address these vulnerabilities and enhance the security of OPC UA systems.

Varadarajan [19] conducts a security analysis of OPC UA in automation systems. The author set up a test environment to simulate an industrial automation system and identify potential vulnerabilities in the OPC UA protocol. The thesis also includes a discussion of detection mechanisms and mitigation strategies to address these vulnerabilities.

The German Federal Office for Information Security (BSI) analyzes the security of the OPC UA protocol [20]. They conduct a survey to assess the implementation of security mechanisms in OPC UA products and the challenges faced by manufacturers. Additionally, the BSI performs a dynamic security analysis and static code analysis to comprehensively identify vulnerabilities across various security aspects of OPC UA. Their findings reveal that many existing products lack essential security features. The BSI urges manufacturers to implement more robust security measures and provide additional training to their clients and users.

IV. Methodology

This section describes the methodological approach used to assess vulnerabilities in OPC UA server implementations. The first subsection describes the overarching strategy used to conduct the vulnerability assessment. It then discusses the considerations that need to be taken into account when planning an assessment environment, while tailoring it to the experiment at hand.

A. General approach

The main objective of this research is to conduct a vulnerability assessment of multiple OPC UA server implementations. To achieve this goal, a methodology, or framework, must be established to guide the assessment process and facilitate understanding. The methodology used in this study is based on well-known frameworks established in the field of cybersecurity, specifically in vulnerability assessment and penetration testing [10, 17]: Open Source Security Testing Methodology Manual (OSSTMM) [21], Penetration Testing Execution Standard (PTES) [22], Information Systems Security Assessment Framework (ISSAF) [23], and Framework for Improving Critical Infrastructure Cybersecurity (FICIC) [24]. While these methodologies offer distinct approaches to vulnerability assessment and penetration testing, they all share a core principle: a systematic approach. This systematic approach can be summarized in the following steps:

1) Information Gathering: This step involves gathering as much information about the system under test as possible and creating an overview about its infrastructure. The information collected can include IP addresses, list of running services and devices, or specific configurations used in systems.

2) Threat Modeling, Vulnerability Identification and Analysis: The aim of this step is to analyze the collected information and use it to create a plan to identify the weak points of the observed system. In addition, the developed plan should be executed and identify as many weak points as possible.

3) Vulnerability Exploitation and Reporting: In this final step, the vulnerabilities identified in the previous step are exploited to assess their impact and potential risks. The results of the exploitation are then documented in a report, which includes a detailed description of the vulnerabilities, their potential impact, and recommended mitigation measures.

This specific approach is chosen for this research due to its comprehensive nature and its ability to cover various aspects of the assessment and testing process.

B. Assessment Environment

To identify all possible vulnerabilities in a system, the presented approach to assess them should be applied on the real system. However, this is not always possible due to the potential risks and consequences of exploiting vulnerabilities in a live system, especially in critical infrastructures [9]. Therefore, a controlled environment is set up to simulate the real system as close as possible (to ensure the results are accurate and reliable) and test the vulnerabilities in a safe and controlled manner. As this research focuses on the vulnerability assessment of OPC UA server implementations, it is not necessary to simulate the entire critical infrastructure. Instead, a controlled environment is set up, which includes the OPC UA server implementations and the tools and frameworks used for vulnerability assessment. This environment offers a network traffic capture without any extraneous traffic, which in turn enables a better training of IDS or IPS systems. The specific environment, OPC UA implementations, and tools used for this research is described in the following section.

V. Implementation

This section details the implementation phase of the OPC UA vulnerability assessment. Here, we delve into the specifics of how the assessment was carried out. First, the test environment that was constructed to simulate a realistic scenario for evaluating OPC UA servers is described. This includes details about the hardware, software, and network configuration used. Following this, the specific OPC UA server implementations that were chosen for the assessment are identified. It explains the selection criteria and provides a brief overview of each server implementation. Next, the various tools and techniques employed during the assessment process are explored. It details the functionalities offered by each tool and how they were used to identifying vulnerabilities within the OPC UA servers. Finally, the step-by-step procedures followed for conducting the vulnerability assessment are outlined. This includes details about the specific activities performed, the order in which they were executed, and the data collected during the process.

A. Environment

Because the vulnerability assessment of OPC UA server implementations requires a controlled environment, a specific setup is needed to simulate the real system and test the vulnerabilities in a safe and controlled manner. As mentioned earlier, it is not necessary to simulate the entire critical infrastructure, but rather a customized environment that includes the OPC UA server implementations and the tools and frameworks used for vulnerability assessment. Therefore, the experimental setup consists of two physical devices and a virtual machine. The first device is a Raspberry Pi 4 Model B with 8GB of RAM running Ubuntu Server 22.04.3 LTS. The Raspberry Pi is used to run the OPC UA server implementations. The second device is a laptop with a 12th Gen Intel© Core™ i7-1260P × 12 and 32GB of RAM running Linux Mint 21.3 Cinnamon. The laptop is used to run the virtual machine and capture network traffic between the Raspberry Pi and the virtual machine. The virtual machine itself has a 12th Gen Intel© Core™ i7-1260P × 4 and 10GB of RAM allocated to it to run Kali Linux 2024.1. The virtual machine is used to run the vulnerability tests against the OPC UA server implementations. For a direct connection between the Raspberry Pi and the virtual machine, the Raspberry Pi is connected to the laptop via an Ethernet cable. This specific configuration is chosen to allow the laptop to capture network traffic between the Raspberry Pi and the virtual machine while running the vulnerability tests. A schematic of the experimental setup is shown in Figure 2.

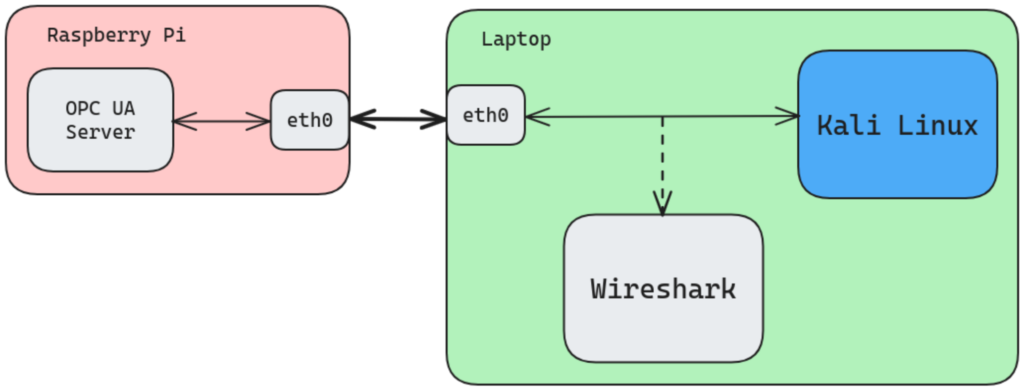


Fig. 2. Schematic of the experimental setup.

B. Server Implementations

The OPC UA server implementations tested in this research are based on different open-source libraries implemented in various programming languages. The base libraries used for the OPC UA server implementations are *open62541* [25] in C/C++, *opcua-asyncio* [26] in Python, *node-opcua* [27] in JavaScript, and *opcua* [28] in Rust. The criteria for this selection are the popularity, active development, and that they are open-source. In order to reduce the effort and complexity of the test setup and to get a first impression of the various libraries before proceeding with further development, the demo servers provided by the libraries are used in this study. A separate server implementation was only developed for the *open62541* library, as this implementation does not provide a demo server.

C. Tools

Since the Raspberry Pi acts as the OPC UA server directly connected to the host laptop running the virtual machine, network scanning or external system information gathering tools are unnecessary for this research. Therefore, the only chosen tools for vulnerability testing in this research are Wireshark [29] (to capture network traffic) and the OPC-UA Exploitation Framework (OEF) by Claroty Team82 [30]. This framework is an open-source penetration testing framework specifically designed for the OPC UA protocol. It offers a diverse set of tools for assessing vulnerabilities and exploiting them. The OEF is chosen for this research due to its wide range of supported OPC UA server implementations, ease of use, and extensive documentation. It provides a wide range of tools for scanning, enumeration and exploitation, making it suitable for testing the security of OPC UA server implementations. The framework includes three different types of attack: Denial of Service (DoS), Remote Code Execution (RCE), and Information Leakage (IL). These are comprised of twelve different attack scenarios, targeting various aspects of the OPC UA protocol, such as session management and data manipulation. In addition to the attack scenarios, the framework includes a set of tools to gather information about an OPC UA server, such as getting diagnostic information about the server or information about specific nodes. More details about the framework and the different attack scenarios can be found in the documentation of the framework [30].

D. Assessment Process

The vulnerability assessment process is carried out as described in section IV-A within the presented environment and using the introduced tools and frameworks. The first step of the process is information gathering. As the environment is set up specifically for this research, the information gathered is limited to the IP addresses of the Raspberry Pi and the virtual machine and the version numbers of the OPC UA server implementations. Therefore, the process can continue with the second step: threat modelling, vulnerability identification, and analysis. Thread models are provided as attack scenarios of the OEF, which are used to identify potential vulnerabilities in the OPC UA server implementations. This means that the vulnerabilities of the servers can now be identified. A simple procedure (described below) has been developed for this purpose, which simplifies the identification process and also documents the whole process extensively.

The first step of this procedure is to start the server on the Raspberry Pi and wait until it is fully operational. Then, an attack scenario is selected from the OEF and executed against the server implementation. After the attack scenario is executed, the server is shut down, and the whole process is repeated for the next attack scenario and sever implementation. For each attack scenario and server implementation, three files are created that serve as documentation: (1) the output of the server called *output-server.log*, (2) the output of the attack scenario called *output-exploit.log*, and (3) the network traffic captured by Wireshark called *wireshark.pcapng*. In the case that an attack scenario does not work as expected, takes longer than expected, or the server crashes, the specific attack is stopped, the server is shut down, and the process is repeated for the next attack scenario and server implementation. The reason for the failure is documented in the files created for the specific attack scenario and server implementation.

This identification procedure is chosen for this research due to its systematic approach and the extensive documentation it provides. The procedure allows for a comprehensive analysis of the results and the potential implications of the vulnerabilities found, which is also part of the second step of the assessment process (see Section IV-A).

The final step of the assessment process is the exploitation of the identified vulnerabilities. This step is crucial to assess the impact of the vulnerabilities and understand the potential risks they pose. But because this research focuses on the identification of vulnerabilities, the exploitation of the vulnerabilities is not part of the assessment process and will therefore not be discussed in this research.

The network traffic generate throughout this assessment is saved in the *wireshark.pcapng* files. The files will be analyzed after the assessment process is finished to determine their usefulness for IDS and IPS systems before using it to train the algorithms.

VI. Results

After following the described assessment process and procedure, the results of the vulnerability testing are obtained. The results presented in Table I show the vulnerabilities found in the different OPC UA server implementations. The presented table only represents the point of view of the attack scenarios. A checkmark means, that the attack scenario executed completely (without any errors), but does not mean that the attack was successful as the server response is not represented in the table. On the other hand, a cross mark means that the attack scenario did not execute completely due to an error (client or server side), the server crashing, or the attack taking longer than expected. Additional to the results of the server implementations against the attack scenarios, the type of attack scenario is also presented in the table.

TABLE I
Vulnerability assessment of OPC UA server implementations.

Attack Scenario	**Attack Type**	**Server implementation**			
		open62541 [25]	***opcua-asyncio [26]***	***node-opcua [27]***	***opcua [28]***
Certificate Infinite Chain Loop	DoS	✗	✗	✗	✓
Chunk Flooding	DoS	✗*	✗*	✗*	✗*
Close Session With Old Timestamp	IL	✓	✓	✓	✓
Complex Nested Message	DoS/IL	✓	✓	✗	✗
Function Call Null Dereference	DoS	✗	✗	✓	✗
Malformed UTF8	RCE	✓	✓	✓	✗
Open Multiple Secure Channels	DoS	✗	✗	✗	✗
Race Change And Browse Address Space	DoS	✓	✗	✗	✗
Thread Pool Wait Starvation	DoS	✗	✓	✓	✗
Translate Browse Path Call Stack Overflow	DoS	✓	✓	✓	✗
Unlimited Condition Refresh	DoS	✗	✗	✗	✗
Unlimited Persistent Subscriptions	DoS	✓	✓	✓	✓
Exploitable attack scenarios		6	6	6	3

DoS: Denial of Service, RCE: Remote Code Execution, IL: Information Leakage
✓: Exploit run completely, ✗: Exploit did not run completely
*: Network capture crashed during the attack scenario.

From Table I several useful information can be depicted. First, all server implementations are vulnerable to DoS and IL attacks, as all of them have at least one attack scenario that can be exploited. Specifically, the attacks *Close Session With Old Timestamp* and *Unlimited Persistent Subscriptions* are successful against all server implementations. In addition, all server implementations except the *opcua* implementation are vulnerable to RCE attacks (see *Malformed UTF8* attack scenario). The reason for this is that the *opcua* server cannot decode the malformed UTF8 string, which results in the server closing the connection with the *BadServiceUnsupported* status. This can be seen in the network traffic captured by Wireshark and the output of the server log. Moreover, the *Chunk Flooding* attack is the only attack scenario that was not fully executed due to Wireshark for all server implementations. This can be traced back to the attack scenario itself, as it floods the Raspberry Pi with messages, which overwhelms Wireshark and causes the laptop to crash.

When looking at the number of exploitable attack scenarios, it can be seen that all implementations have six exploitable attack scenarios, except the *opcua* implementation, which only has three exploitable attack scenarios. This shows that the *opcua* implementation is less vulnerable to cyberattacks compared to the other implementations.

The vulnerability assessment process yields a comprehensive understanding of potential weaknesses within the tested OPC UA server implementations. However, the captured network traffic extends the value of this research beyond the identification of vulnerabilities. Going through the data, it is clear to see, that it will be a valuable asset in training machine learning algorithm to detect these attacks.

VII. Discussion

The results of the vulnerability assessment show that all tested OPC UA server implementations are vulnerable to various types of attacks, ranging from DoS to RCE and IL. Furthermore, the results in Table I indicate, that the least vulnerable implementation is the *opcua* [28] implementation, which only has three exploitable attack scenarios. This is a result of testing the demo servers of the libraries, as they are set up with different configurations and functionalities, and therefore respond differently to the attack scenarios. The demo servers are only configured to show the basic functionality of the respective library. They are not configured to be secure and not every functionality of the library is included nor is every specification of the protocol implemented. Those differences in the configuration and functionality of the demo servers are the reason for the different results in the vulnerability assessment. In addition, other factors such as the programming language, how the functionalities are implemented or the implementation priority of protocol specifications play an important role in the vulnerability assessment. These are reasons, why the exploitable attack scenarios vary between the different server implementations.

Beyond identifying vulnerabilities in OPC UA server implementations, this research also aimed to generate valuable traffic data for training machine learning algorithms employed in IDS and IPS Systems. The captured traffic encompasses both normal and malicious behavior, providing a rich dataset for algorithm development. By training on this comprehensive data, machine learning models can be fine-tuned to effectively recognize and thwart future cyberattacks targeting OPC UA systems. For instance, the captured traffic data containing the successful *Close Session With Old Timestamp* and *Unlimited Persistent Subscriptions* attacks can be used to train machine learning models to identify similar attack patterns in real-time network traffic. This can significantly improve the effectiveness of IDS and IPS in detecting and preventing DoS attacks against OPC UA servers. Similarly, the data from the *Malformed UTF8* attack scenario, which exploited an RCE vulnerability in all implementations except *opcua*, can be used to train machine learning models to detect such malformed data packets. This can help prevent RCE attacks that attempt to inject malicious code into vulnerable OPC UA servers. Overall, the generated traffic data plays a crucial role in developing robust machine learning-based IDS/IPS solutions that can effectively safeguard OPC UA systems from cyberattacks.

VIII. Conclusion and Future Work

This research conducted a comprehensive vulnerability assessment of multiple OPC UA server implementations, leveraging established frameworks like OSSTMM, PTES, ISSAF, and FICIC. The assessment environment was carefully controlled to simulate a real system while enabling safe and controlled vulnerability testing. The tested implementations stemmed from various open-source libraries written in different programming languages. Wireshark and the OPC-UA Exploitation Framework by Claroty Team82 served as the tools for vulnerability assessment.

The results revealed vulnerabilities across all tested OPC UA server implementations, encompassing DoS, RCE, and IL attacks. The *opcua* library-based implementation exhibited the least vulnerability among those assessed. These findings emphasize the critical need for prioritizing security throughout the development lifecycle of OPC UA servers. Developers should integrate secure coding practices, conduct regular security testing, and maintain updated libraries and frameworks to minimize vulnerabilities. System administrators also play a crucial role by understanding the security risks associated with OPC UA servers and implementing appropriate safeguards like network segmentation, IDS or IPS, and firewalls.

Beyond the vulnerability assessment, this research has a significant additional accomplishment: generating valuable traffic data for training machine learning algorithms used in IDS and IPS systems. The captured traffic data includes both normal and malicious behavior, providing a rich dataset for developing robust machine learning models. These models can be trained to effectively identify and thwart future cyberattacks targeting OPC UA systems.

The vulnerability assessment was conducted in a controlled environment, and expanding it to encompass a broader range of OPC UA server implementations within real-world industrial automation environments would be a valuable next step. The generated traffic data presents a unique opportunity to develop and evaluate machine learning-based IDS or IPS solutions specifically tailored to OPC UA security. Furthermore, investigating more advanced attack vectors and evasion techniques employed by malicious actors would provide deeper insights into potential threats. Examining the performance characteristics of various OPC UA server implementations under stress conditions (performance profiling) could also be an area for future research. This analysis would help system administrators select and configure OPC UA servers for optimal performance and security in real-world deployments.

By addressing these limitations and pursuing further research avenues, we can continuously improve the security posture of OPC UA and safeguard critical infrastructure systems from evolving cyber threats. The integration of machine learning-based IDS or IPS solutions, informed by the traffic data generated in this research, holds immense promise for enhancing the overall security landscape of OPC UA communication.

ACKNOWLEDGEMENT

The presented work is part of the research project *KRITIS Scalable Safe and Secure Modules* (KRITIS³M), which is funded by the Project Management Jülich (PtJ) and the German Federal Ministry for Economic Affairs and Climate Action (BMWK) under funding code 03EI6089A.

REFERENCES

[1] C. R. Team. "Check Point Research Reports a 38% Increase in 2022 Global Cyberattacks," Check Point Blog. (Jan. 5, 2023), [Online]. Available: https://blog.checkpoint.com/2023/01/05/38-increase-in-2022-global-cyberattacks/ (visited on 05/23/2024).

[2] Bundesamt für Sicherheit in der Informationstechnik, "Zweites Gesetz zur Erhöhung der Sicherheit informationstechnischer Systeme," *Bundesgesetzblatt Teil I*, no. 25, p. 1122, May 27, 2021. [Online]. Available: http://www.bgbl.de/xaver/bgbl/start.xav?startbk=Bundesanzeiger_BGBl&jumpTo=bgbl121s1122.pdf (visited on 05/23/2024).

[3] E. Antoni. "Kritische Infrastrukturen absichern - Kickoff des Fördervorhabens KRITIS³M - LaS³." (May 2, 2023), [Online]. Available: https://www.las3.de/kickoff-kritis%c2%b3m/ (visited on 05/23/2024).

[4] OTH Regensburg. "Für sichere Energie- und Wasserversorgung." (May 23, 2024), [Online]. Available: https://www.oth-regensburg.de/news/detailansicht/fuer-sichere-energie-und-wasserversorgung (visited on 02/22/2024).

[5] Bundesamt für Sicherheit in der Informationstechnik. "Bundesamt für Sicherheit in der Informationstechnik," Bundesamt für Sicherheit in der Informationstechnik. (Jun. 3, 2024), [Online]. Available: https://www.bsi.bund.de/DE/Home/home_node.html (visited on 05/25/2024).

[6] European Union Agency for Cybersecurity. "ENISA." (2024), [Online]. Available: https://www.enisa.europa.eu/ (visited on 05/25/2024).

[7] National Institute of Standards and Technology. "National Institute of Standards and Technology," NIST. (May 21, 2024), [Online]. Available: https://www.nist.gov/ (visited on 05/25/2024).

[8] OPC Foundation. "Security Bulletins," OPC Foundation. (2024), [Online]. Available: https://opcfoundation.org/security-bulletins/ (visited on 05/13/2024).

[9] A. Bartusiak, M. Kühne, O. Nitschke, J. Lässig, S. Nicolai, and P. Bretschneider, "First step into automation of security assessment of critical infrastructures," *Sustainable Energy, Grids and Networks*, vol. 36, p. 101 139, Dec. 1, 2023, ISSN: 2352-4677. DOI: 10.1016/j.segan.2023.101139. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352467723001479 (visited on 05/14/2024).

[10] M. Alhamed and M. M. H. Rahman, "A Systematic Literature Review on Penetration Testing in Networks: Future Research Directions," *Applied Sciences*, vol. 13, no. 12, p. 6986, 12 Jan. 2023, ISSN: 2076-3417. DOI: 10.3390/app13126986. [Online]. Available: https://www.mdpi.com/2076-3417/13/12/6986 (visited on 05/14/2024).

[11] OPC Foundation. "Classic," OPC Foundation. (2024), [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-classic/ (visited on 05/23/2024).

[12] OPC Foundation. "History," OPC Foundation. (2024), [Online]. Available: https://opcfoundation.org/about/opc-foundation/history/ (visited on 05/23/2024).

[13] OPC Foundation. "Unified Architecture," OPC Foundation. (2024), [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua/ (visited on 05/23/2024).

[14] O. Foundation. "UA Companion Specifications," OPC Foundation. (2024), [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua/ua-companion-specifications/ (visited on 05/28/2024).

[15] O. Foundation. "UA Part 4: Services," OPC Foundation. (Dec. 13, 2023), [Online]. Available: https://opcfoundation.org/developer-tools/documents/view/161 (visited on 05/28/2024).

[16] O. Foundation. "UA Part 5: Information Model," OPC Foundation. (Dec. 13, 2023), [Online]. Available: https://opcfoundation.org/developer-tools/documents/view/162 (visited on 05/28/2024).

[17] K. U. Sarker, F. Yunus, and A. Deraman, "Penetration Taxonomy: A Systematic Review on the Penetration Process, Framework, Standards, Tools, and Scoring Methods," *Sustainability*, vol. 15, no. 13, p. 10 471, 13 Jan. 2023, ISSN: 2071-1050. DOI: 10.3390/su151310471. [Online]. Available: https://www.mdpi.com/2071-1050/15/13/10471 (visited on 05/14/2024).

[18] T. Ivanova, Y. Belev, and I. Batchkova, "Cybersecurity of OPC ua based cyber-physical systems," *Industry 4.0*, vol. 6, no. 6, pp. 204–207, 2021. [Online]. Available: https://stumejournals.com/journals/i4/2021/6/204 (visited on 03/31/2024).

[19] V. Varadarajan, *Security Analysis of OPC UA in Automation Systems for IIoT*. 2022. [Online]. Available: https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-320100 (visited on 03/31/2024).

[20] Bundesamt für Sicherheit in der Informationstechnik, "OPC-UA security analysis," no. 3, Jun. 1, 2022.

[21] P. Herzog, *OSSTMM 3 - The Open Source Security Testing Methodology Manual*, ISECOM, 2024. [Online]. Available: https://www.isecom.org/OSSTMM.3.pdf (visited on 05/26/2024).

[22] C. Nickerson *et al.* “The Penetration Testing Execution Standard.” (2024), [Online]. Available: http://www.pentest-standard.org/index.php/Main_Page (visited on 05/26/2024).

[23] B. Rathore *et al.*, “Information Systems Security Assessment Framework,” 2005. [Online]. Available: https://untrustednetwork.net/files/issaf0.2.1.pdf (visited on 05/28/2024).

[24] M. P. Barrett, “Framework for Improving Critical Infrastructure Cybersecurity Version 1.1,” *NIST*, Apr. 16, 2018. [Online]. Available: https://www.nist.gov/publications/framework-improving-critical-infrastructure-cybersecurity-version-11 (visited on 05/26/2024).

[25] *Open62541/open62541*, open62541, Dec. 15, 2013. [Online]. Available: https://github.com/open62541/open62541 (visited on 05/22/2024).

[26] oroulet, A. Heine, A. Rykovanov, J. Dr. Denis, and mdcb, *FreeOpcUa/opcua-asyncio*, version 1.1.0, Free OPC-UA Library, Feb. 15, 2025. [Online]. Available: https://github.com/FreeOpcUa/opcua-asyncio (visited on 05/22/2024).

[27] Sterfive, *Node-opcua/node-opcua:* 2022. [Online]. Available: https://github.com/node-opcua/node-opcua (visited on 05/22/2024).

[28] locka99, *Locka99/opcua*, Jan. 1, 2017. [Online]. Available: https://github.com/locka99/opcua (visited on 05/22/2024).

[29] Wireshark Foundation. “Wireshark · Go Deep,” Wireshark. (2024), [Online]. Available: https://www.wireshark.org/ (visited on 05/27/2024).

[30] S. Brizinov, U. Katz, N. Moshe, V. Brizinov, and A. Preminger, *Claroty/opcua-exploit-framework*, Claroty, Jun. 4, 2023. [Online]. Available: https://github.com/claroty/opcua-exploit-framework (visited on 05/22/2024).

Assessment of a MACsec-based Security System for Use in Critical Infrastructure Communication

Lukas Füreder, Prof. Dr. Jürgen Mottok
Technical University of Applied Sciences Regensburg (OTH)
Laboratory for Safe and Secure Systems (LaS³)
Regensburg, Germany
{lukas.fuereder, juergen.mottok}@oth-regensburg.de

Abstract

This paper investigates the integration of Media Access Control Security (MACsec) into the communication of critical infrastructure, specifically within power grid applications, such as Substation Automation Systems (SAS) using the IEC 61850 standard. Building on the principles of both standards, this study aims to determine if MACsec can meet the security and performance requirements set by IEC 62351 for power system communications.

Furthermore, a test environment containing a number of Intelligent Electronic Devices (IEDs) supporting communication compliant to all IEC 61850 message types is established to evaluate this integration. The results of the measurements executed in this environment indicate that MACsec could secure all types of messages, such as Manufacturing Message Specification (MMS), Sampled-Value (SV) and Generic Object Oriented Substation Events (GOOSE), within the required time periods without significant delays. Even with additional encryption activated in the cipher suite, the resulting transmission times are well below the required times. This suggests that MACsec can enhance the security goals for industrial communication by providing confidentiality in addition to the already mandated assurance of authenticity and integrity to all messages without compromising performance.

Only the requirement for end-to-end security cannot be met by MACsec in this configuration, as the security system re-encrypts with every hop of the transmission. For this reason, we propose a hybrid approach of Transport Layer Security (TLS) and MACsec as part of future work.

***Keywords*— MACsec, IEC61850, IEC62351, GOOSE, Secure Communication**

I. Introduction

The steady progress of digitalization is creating many new opportunities for society, business and science. However, this increasing connectivity especially among system relevant institutions and companies is also creating new challenges and potential threats. Companies classified as critical infrastructure, for example water supply facilities, power plants and their corresponding distribution systems, can constitute an attractive target for cyber attacks, which could disrupt the supply of basic resources to entire countries. As a result, laws such as the Network and Information Security Act (NIS-2) [1] of the European Union or the IT Act 2.0 [26] enforced by the German Federal Office for Information Security (BSI) demand a unified level of cybersecurity for these entities. In these regulations, the councils prescribe that the companies must adhere to information security standards, which mandate the current requirements for secure communication and cryptography [3, p. 9]. Furthermore, the extension of the IT Act 2.0 dictates that these companies are obliged to provide proof of compliance with the security requirements over a two-year period [26, §11 (1e)]. This decision is intended to ensure the future operability of the security systems with respect to adapting changes of the latest technologies.

The main objectives of these security standards are the assurance of the security goals authenticity, integrity and confidentiality for applications in critical infrastructure [26, §2 (13)]. Organizations such as the International Electrotechnical Commission (IEC) or the Institute of Electrical and Electronics Engineers (IEEE) develop standards that specify the implementation of these abstract security goals. This paper evaluates the currently established implementation of protection systems securing communication in Substation Automation Systems (SASs) and thereby provides a brief overview of the communication standards used in these facilities. Following this, we propose a Media Access Control Security (MACsec) based security system and evaluate whether this implementation fulfills the requirements of the security goals specified in the IEC 62351 security standards.

The further course of the paper is structured as follows: Chapter II clarifies the technical background of this paper and thereby provides a general overview of the IEC 61850 communication standard and the associated message types. Building on this, the further part of this chapter presents the current state of message security mandated by the IEC 62351 security standard. Following this, a brief introduction into the MACsec security standard is provided, which presents the relevant features used in the implementation later on. Chapter III displays relevant information presented by related work assessing the current state of technology in this topic. In the further course of the paper, Chapter IV explains the test setup used to measure the performance of the MACsec-based security system. In chapter V the data gathered is evaluated and placed in the context of the mandated security requirements.

II. Background

A. Overview of the IEC 61850 Standard

Among other standards used for communication in industrial applications, power systems primarily utilize the IEC 61850 standard [6], which is published and maintained by the International Electrotechnical Comission (IEC) [18]. This standard specifies the transmission of diagnostic information, measurement values or control signals among devices structured in a hierarchical three level architecture [22], as displayed in Figure 1. The major advantage with this type of communication consists of the object-oriented data structure defined in this standard, which makes the integration of various components developed by different vendors possible [5, p. 5643].

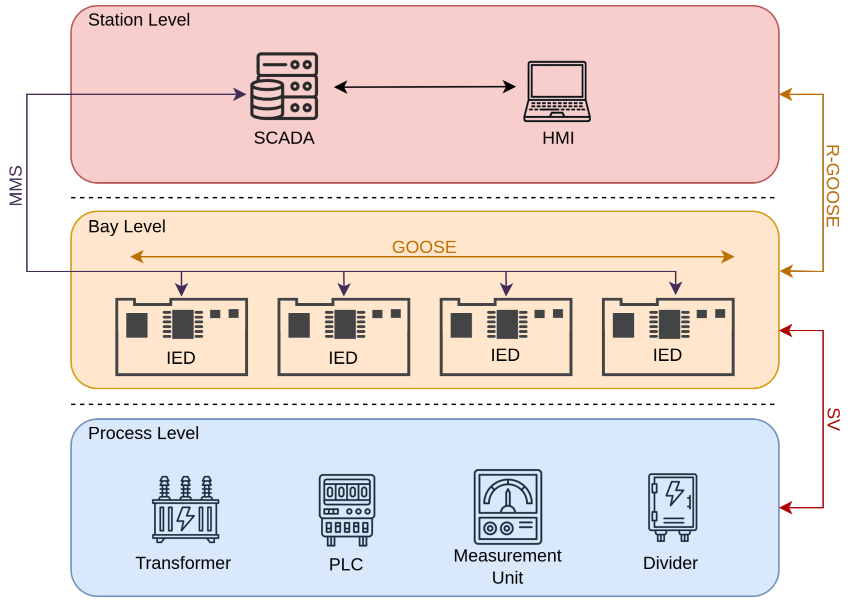


Figure 1: Overview of the three architecture levels in IEC 61850 [22]

Closest to the power lines, the *Process Level* contains devices tasked with the actual power regulation. Examples for these are: transformers, circuit breakers, Programmable Logic Controllers (PLCs) and measurement units [22]. Upon configuration, Process Level components periodically publish measurement information to all subscribing communication partners in the Bay Level via Sampled-Value (SV) packets [23]. This communication involves LAN-internal multicast packets, which take place exclusively on the second layer (Data Link Layer) of the Open System Interconnection (OSI) model [23].

The *Bay Level* above contains the Intelligent Electronic Devices (IEDs), each of which represents an indepenent logical component in the substation [8, p. 29]. The IEDs gather the measurement data from the Process Level and initially process them. The resulting information is then communicated through Manufacturing Message Specification (MMS) packets to other IEDs and the Station Level components [11, p. 44]. Simultaneously, the IEDs receive control signals from the Station Level, which are also transmitted via MMS packets [15]. As the Station Level components are not necessarily located in the same LAN as the IEDs, the MMS messaging is implemented through TCP packets on the fourth layer (Transport Layer) of the OSI model [11, p. 45]. In addition to the MMS messaging, the IEDs use Generic Object Oriented Substation Events (GOOSE) to send time-critical information to surrounding IEDs. Similar to SV, GOOSE messages are implemented on layer 2 of the OSI model through multicast Ethernet packets in the LAN [4].

The devices located in the *Station Level* of the architecture are responsible for controlling the SAS. This is achieved through MMS packets addressed to specific IEDs and the presentation of the processed information in graphical illustrations to the user [22]. For this, the Station Level components typically consist of a Supervisory Control and Data Acquisition (SCADA) component and a Human-Machine-Interface (HMI). As displayed in Figure 1, it is possible to transmit GOOSE messages to the Station Level components. For this, the IEC 61850-90-5 standard [12] defines a routable version of the layer 2 GOOSE frame. For this purpose, the GOOSE packets are extended by adding network and transport layer headers to form a UDP packet, which can be routed through multiple hops in a Wide Area Network (WAN) [24].

B. Message Security according to IEC 62351

Building on the IEC 61850 message types described in Chapter II-A, the IEC 62351 standard [13] dictates security goals and requirements for cybersecurity solutions. In order to evaluate the proposed MACsec solution for industrial applications, we need to assess the proposed security functions according to the demands of this standard.

With regard to MMS messages, the standard prescribes the protection of authenticity, integrity and confidentiality. IEC 62351-4 directly specifies certificate-based TLS to achieve these security goals [25]. Furthermore, the standard subdivides the assertion of the security requirements according to the layers of the OSI reference model. The upper three layers are summarized in the application profile and the lower four layers in the transport profile [25]. Based on this, the standard specifies that the security system shall verify the authenticity of the communication partner and the integrity of the transmitted messages during the handshake phase of the transport profile [5]. Following this, the system shall provide confidentiality for the outgoing messages in the data transfer phase [25]. With respect to the upper three layers, the standard specifies two possible implementations in the application profile: peer-to-peer security and end-to-end security [5]. The primary difference between them consists in the data origin authentication and message integrity checks, which are only verified during the association establishment in the peer-to-peer implementation, whereas the end-to-end implementation also ensures them in the following data transfer [5].

For GOOSE messages, the standard argues that the strict real-time requirement of a maximum of 3 ms [4] outweighs the security requirements and, for this reason, state that security measures which affect the transmission rates are not acceptable [16]. Building on this, the IEC 62351 standard advises against the encryption of GOOSE messages and only proposes the use of digital signatures to verify the authenticity of the GOOSE publisher and the integrity of the message. As similar restrictions arise for SV messages, the standard equally advises the usage of digital signatures for SV packets [5].

C. Fundamentals of the MACsec Security Standard

MACsec represents an information security standard which protects messages on the second layer (Data Link Layer) of the OSI model. In contrast to security standards operating in higher layers of the TCP/IP stack (e.g. TLS), which provide end-to-end encryption, MACsec verifies the confidentiality, integrity and authenticity of a packet within each hop of the transmission [19]. However, this lower layer implementation close to the PHY enables MACsec to secure communication which takes place exclusively on layer 2 (e.g. SV & GOOSE). The following paragraph explains the most important aspects of the MACsec standard, which are responsible for ensuring the authenticity, integrity and confidentiality of the transmitted packets.

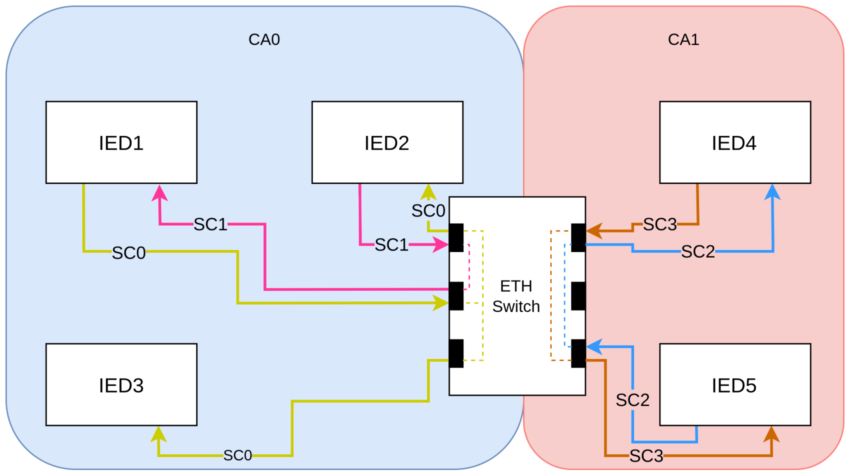


Figure 2: Schematic representation of the MACsec entities [14]

As displayed in Figure 2, the communicating devices are initially grouped into Connectivity Associations (CAs), which represent the logical separation of secured communication areas [14, p. 35]. Each member of a CA possesses the associated Connectivity Association Key (CAK), which is later used to generate the individual session keys. Similar to other encryption systems, the CAK acts as a shared secret between the individual parties and is therefore used to verify the authenticity of other devices in the same CA. In the IEEE 802.1AE standard, the distribution of the CAK among the participants is only specified as a brief overview of the established methods [14, p. 230]. In our case, we utilize pre-shared keys to configure the CA in the test setup explained in Chapter IV. Within a CA, the connections between the communicating devices are referred to as Secure Channels (SCs). As displayed in Figure 2, an SC is a unidirectional connection from a transmitter to one or more receivers. Each SC can be identified by the Secure Channel Identifier (SCI), which is added to the MACsec specific field in the secured frame

[14, p. 43]. During transmission over an SC, the packets are sent within Security Associations (SAs). These are time intervals in which a single Secure Association Key (SAK) is valid [14, p. 44]. The SAK is the session key, which is derived from the previously described CAK and is only valid for up to $(2^{32} - 1)$ packets, after which a new SAK needs to be generated [14, p. 66]. To be able to monitor the number of transmitted packets in an SA, the MACsec frame contains a packet number field, which is incremented with each subsequent packet. This field additionally ensures that no replay attacks can be carried out on the network [14, p. 145].

To ensure authenticity, confidentiality and integrity of the transmitted frames, MACsec utilizes Authenticated Encryption with Associated Data (AEAD) cipher suites. These encryption systems typically consist of a symmetric block cipher and a Message Authentication Code (MAC) generator [20]. The first entity of which provides the option to encrypt the payload of the message, while the second entity simultaneously generates a MAC over the entire message [4]. For usage, the IEEE 802.1AE standard specifies four variations of the Advances Encryption System in Galois/Counter Mode (AES-GCM) [14, p. 143ff].

III. Related Work

To assess the operating principle of a MACsec-based security system in IEC 61850 compliant communication, it is necessary to understand both the working method of the communication inside a substation as well as the corresponding functionality of the MACsec security standard. The following related work display these important aspects and are therefore relevant for the implementation of an experimental setup for MACsec secured industrial communication.

Mackiewicz [18] describes the overall usage of the IEC 61850 protocol by illustrating key features as well as the general aspects of IEC 61850 compliant communication. Since this standard represents a core part of the communication inside of power grid systems, it is vital to understand the corresponding aspects such as communication paths, model structures or data addressing in order to design a representative test environment.

Hussain *et al.* [5] published a paper assessing the IEC 62351 standard and its security mechanisms towards IEC 61850 compliant messaging. The publication initially describes the basic values and security goals of the safety standard and, building on this, which attacks can potentially be carried out on IEC 61850 messages to manipulate the internal workings of a SAS. At this point, the paper primarily focuses on the Ethernet-based message types GOOSE and SV and the associated decision not to encrypt them due to strict real-time delivery requirements.

Lackorzynski *et al.* [17] proposed modifications of the IEEE 802.1AE standard to improve MACsec for usage in industrial applications. In particular, the fragmentation of Ethernet frames was considered. This procedure is necessary, if messages exceed the Maximum Transmission Unit (MTU) and are thus possibly discarded by the recipient of the message. The presented implementation ensures the adherence to the MTU and splits messages into multiple frames if it is exceeded. Additionally, the authors discuss the usage of different cipher suits instead of the AES-GCM 128/256 specified in the MACsec standard. The evaluation of their study shows that the ChaCha20-Poly1305 cipher is a promising alternative for industrial applications.

Moreira *et al.* [19] evaluate various approaches to introduce cybersecurity in SASs. Initially, a brief outline of the communication structures in substations is presented. Building on this, various established security approaches are explained and evaluated based on the protection objectives of the IEC 62351 standard. The authors also point out possible implementation problems, such as incompatibilities between the security systems and the communication protocols or the handling of redundant packets inside ring-topology networks. In the further course of the paper, the authors propose the idea of MACsec-based communication security in SASs and discuss the possible advantages and challenges that arise with it.

Hussain *et al.* [4] analyzed possible GOOSE security implementations based on their preceding review of the IEC 62351 standard [5]. Especially concerning the decision to abstain from implementing confidentiality in GOOSE messages, the authors argue that the critical payload of these messages demand encryption to provide efficient protection against attacks. However, in order to meet the real-time requirements of the protocol, they suggest replacing the RSA signature with encryption using an AEAD cipher. In the further course of the paper, they compare encryption and signature times between different AEAD ciphers and conclude based on the measurement results that these encryption systems pose a promising solution, which provides the opportunity to encrypt the message while simultaneously meeting the 3 ms timing requirement.

Building on the theoretical proposition of Moreira *et al.* [19], we formulate our evaluation of MACsec for use in substations and other power systems based on the IEC 61850 standard. Along with this, we consider the findings of Hussain *et al.* [5] in relation to the proposed modifications of the security goals for Ethernet-based messaging in IEC 62351 for the implementation of our MACsec test environment. From this, our experimental setup enables us to discuss the advantages and disadvantages of MACsec in comparison with the security goals of the IEC 62351 standard as well as the timing requirements of the IEC 61850 standard.

IV. Implementation

To display the communication in a SAS, we implement a test environment consisting of three Bay Level components. As the IEDs take part in all forms of message exchange inside the substation architecture, they are perfectly suitable for testing the communication in conjunction with MACsec. In order to ensure the reproducibility of this study, we decided to use the Raspberry Pi 4 as hardware platform for all devices in the setup. With respect to the software used in the applications, we utilize the open-source library libiec61850[1] to establish the different communication types and data structures of the IEC 61850 standard. To integrate MACsec into the communication, we utilize the MACsec Linux kernel module[2], which establishes a configurable virtual interface on top of an existing network interface [2]. For the implementation of the time measurement in Chapter IV-C, we integrate the WiringPi[3] library into the project. This enables us to access peripherals of the hardware platform and results in precise time measurement across several communication participants.

The remainder of this chapter proceeds as follows: Chapter IV-A initially explains the overall structure of the test environment. Chapter IV-B describes the optionally activatable MACsec configuration and the corresponding entities of the security standard. Chapter IV-C elaborates on the subsequent test executions and the transmission time measurement.

A. *Structure of the Test Environment*

As displayed in Figure 3, we configure IED1 as a publishing server and IED2 and IED3 as subscribing clients. Furthermore, the implementation of IED1 contains an XML data structure compliant to the specification of the Substation Configuration Language (SCL) in IEC 61850-6 [7]. This file contains the communication and processing information of the IED itself [18].

In our case, we integrate a measurement unit (MMXU) [10, p. 268] and a control unit (LLN0) [10, p. 164] into the SCL file of IED1. During runtime, IED1 continuously populates the data points of the measuring unit with sampled values of a sinusoidal function. In addition to the actual measurement, each sampled value contains a time stamp and a quality index [9, p. 61ff]. These values can then be requested by IED2 and IED3 via MMS messages.

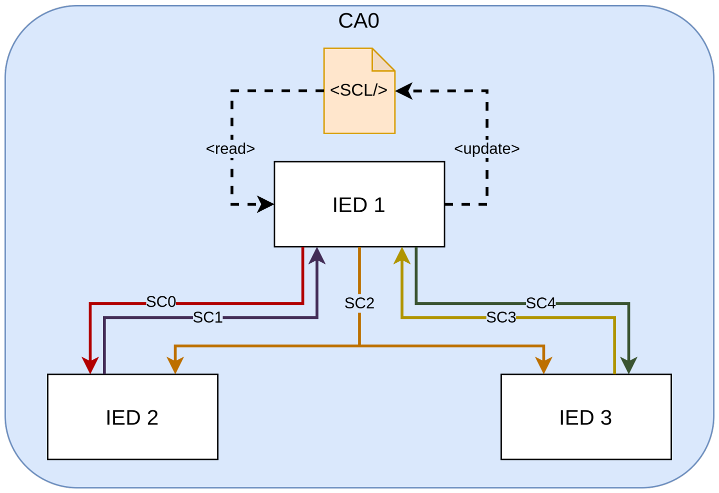


Figure 3: Component Structure of the Test Setup

In addition to the TCP request-response communication via MMS messages, IED1 periodically publishes GOOSE and SV messages in a configurable interval containing the current value of the measurement. The configuration of the GOOSE messages, which contains the corresponding multicast MAC address, VLAN ID and VLAN priority, are equally stored in the control unit of the SCL file [11, p. 189]. For the transmission of the SV packets, an Application Protocol Data Unit (APDU) is configured during the initialization phase of the server. This object contains the floating-point value of the latest measurement and a message time stamp. As the application progresses, the content of the APDU is periodically updated and published. On the subscriber side, the subscription to the corresponding events (in this case GOOSE and SV) is implemented in software in the form of asynchronous handler functions, which subsequently log the incoming information.

B. *MACsec integration in the Test Environment*

Building on the general structure of the test environment described in Chapter IV-A, the implementation of MACsec into the test setup can now be explained. Initially, we integrate all devices in the same CA by distributing a pre-shared CAK. Based on this key, the various SCs and SAs can be established as shown in Figure 3. For bidirectional MMS communication between two IEDs, one SC is required for each communication direction. Using the example of packet exchange between IED1 and

[1]source: https://libiec61850.com/
[2]source: https://github.com/torvalds/linux/blob/master/drivers/net/macsec.c
[3]source: https://github.com/WiringPi/WiringPi

IED2, we have configured the secure channels SC0 and SC1 for this purpose. Each of these channels derives its own SAK from the CAK and establishes a secure connection for the duration of the SA. The establishment of SC3 and SC4 is carried out identically for MMS communication between IED1 and IED3.

With regard to the secure publication of GOOSE and SV packets, we established an additional SC (displayed in Figure 3 as SC2), which is connected to both IED2 and IED3. This configuration enables us to multicast the messages while simultaneously securing the payload and ensuring authenticity and integrity. Since the configuration of this test environment only allows the publication of GOOSE and SV packets originating from IED1, we only need one SC to secure them with MACsec. Consequently, if IED2 and IED3 are also meant to send secure multicast messages, two additional SCs must be configured.

C. *Definition of the Test Procedures*

To evaluate whether this MACsec implementation fulfills the time requirements of the IEC 61850 standard, we conduct a measurement of the transmission times for each of the different message types. In order to achieve a precise time measurement, we expanded the implementation of the server and the client application to toggle a General Purpose Input/Output (GPIO) pin during transmission and reception of the associated frame type. By using an external logic analyzer, the times between the voltage level shifts can be measured. Since this measurement is performed by an external device, we do not need clock synchronization between the communication participants, which significantly reduces the implementation efforts of the measurement.

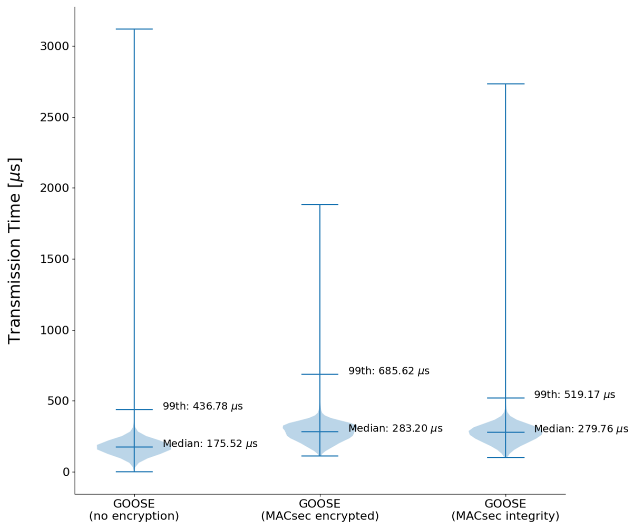
V. Evaluation

Based on the implementation design explained in Chapter IV, a number of transmission time measurements can be executed. We compare the times in three different configurations in order to be able to investigate the temporal influence of the security module. The first step is to measure the transmission times without the security module active, followed by MACsec frame protection with integrity check and encryption and lastly solely with MACsec integrity check.

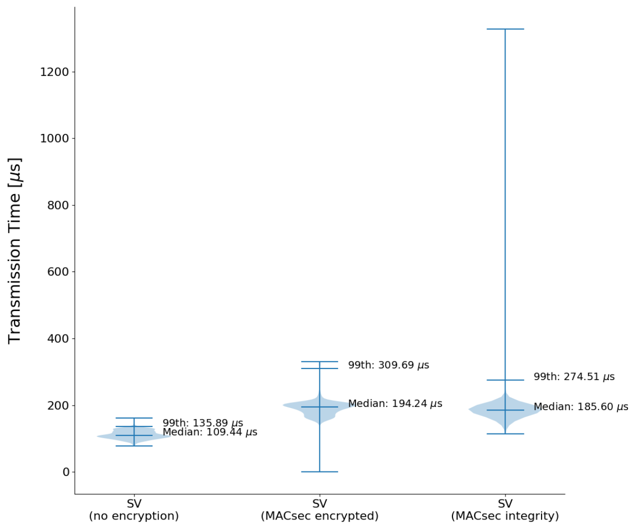
These measurements are carried out in the following chapters for all IEC 61850 message types. The further course of this chapter is structured as follows: Chapter V-A illustrates the measurement results for GOOSE and SV messages and Chapter V-B presents the results for MMS messages.

A. *Temporal Influence of MACsec on SV and GOOSE messages*

For the GOOSE and SV measurement, IED1 was configured in the test setup so that the corresponding packets are published periodically at intervals of 500 ms. In the next step, the temporal delay to the handler function of the client side was measured. This measurement was then repeated 1000 times without additional computational load for the three illustrated configurations and is therefore under ideal conditions. The results of these measurements are displayed in Figures 4a and 4b.



(a) Time comparison of a single GOOSE transmission



(b) Time comparison of a single SV transmission

In addition to the mean transmission time, we display the 99^{th} percentile in all plots. This value is used as comparative value for the absolute transmission time in the further course of this paper. As can be seen here, the 99^{th} percentile for both Ethernet-based packets is transmitted in well below the IEC 61850 standard specification of 3 ms [21] regardless of the chosen configuration. Furthermore, securing the packets results in an extension of the transmission time of approximately 100 μs for both message types. It can be seen that activating the encryption in the AEAD cipher leads to a minimal increase in transmission time. This is likely due to the necessary rearrangement of the data payload into the Ethernet frame, which is not needed if only integrity checks are performed. Overall, the change in transmission times between these two configurations is nevertheless minimal, at around 6 μs for both message types.

B. Temporal Influence of MACsec on MMS messages

For the MMS measurement, IED1 was configured to only perform the periodic update of the data points in the measurement unit of the internal SCL data structure. During this process, the time required for an MMS request-response sequence is measured on the client side. Since this message exchange involves two MMS messages, the resulting time is divided by two in the next step to determine the average transmission time. Identical to Chapter V-A, the measurement is then repeated 1000 times for all three configurations. The results of these measurements are displayed in Figure 5.

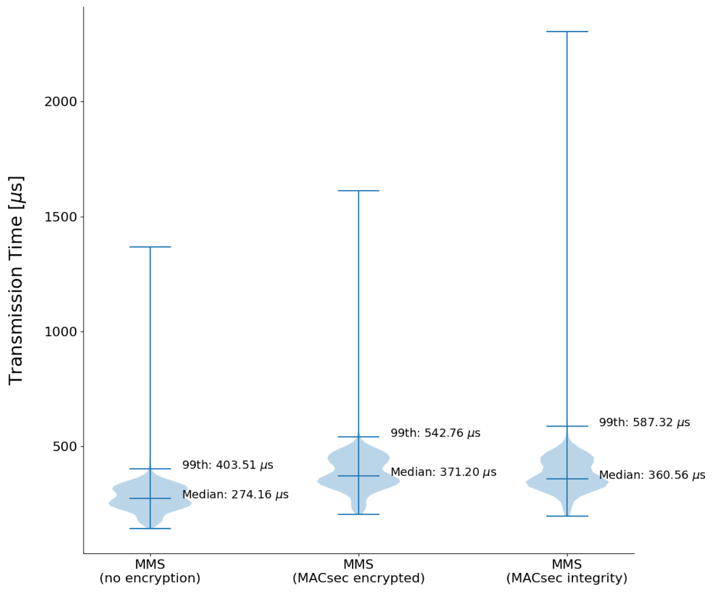


Figure 5: Time comparison of a single MMS transmission

As displayed in Figure 5, the time measurements for MMS messages vary only marginally when compared to the transmission times of the Ethernet-based packets discussed in chapter V-A. This delay is likely due to the overhead caused by the additional processing in the higher layers of the OSI stack. Moreover, the time measurement for MMS transmission contains the application internal management of the SCL data structure in IED1, which further impacts the absolute transmission time of the measurements. However, the trend in the variance between the different test configurations is consistent with the other measurements, as the additional processing time for the MACsec protection of the frame is set to approximately 100 μs. Analog to GOOSE and SV, the IEC 61850 standard specifies a timing requirement for MMS packets of 100 ms [21]. As demonstrated in this measurement, the transmission time of MMS messaging with an active MACsec security module is well below the required time period.

VI. Conclusion

The aim of this paper was to investigate the extent to which the integration of MACsec in industrial communication would provide a viable alternative to the currently established security systems. In order to be able to evaluate this correctly, we implemented a test environment consisting of three IEDs, which can be configured to exchange the desired IEC 61850 messages.

Although the measurement data collected in this test setup does not provide the scope to make a general statement about compliance with the time requirements, it does provide an initial insight into the realization of a MACsec-based security system. Going further, we evaluated, based on our measurement results, whether the timing and security requirements of the IEC 61850 and IEC 62351 standards can be considered fulfilled.

With regard to the time requirements of the messages, Chapter V shows that the 99^{th} percentile of all packets is transmitted in well below the required time periods. In addition to this, the measurements prove that the optionally activatable encryption of the AEAD cipher does not lead to a major increase in computational overhead. Therefore, confidentiality can be added to the security goals for all message types, regardless of the OSI model layer on which they operate.

Only the IEC 62351 requirement mandating that a complete end-to-end security model for power systems shall be implemented [5] cannot be fulfilled by MACsec. Due to the underlying architecture of this security standard, MACsec provides encryption of Ethernet-based packets, but must therefore be re-encrypted at each hop of the transmission. Other security standards such as TLS would fulfill this requirement, as they operate based on the TCP frame structure and can therefore be routed without a decryption of the message. However, simultaneously, packets below layer 4 of the OSI stack cannot be processed with TLS, which again would result in GOOSE and SV packets being transmitted unencrypted if implemented in a SAS. For this reason, we propose a hybrid implementation of both standards in a SAS for future work. In this scenario, MACsec could be used for LAN-internal communication security, while TLS provides the protection from a gateway component to external communication partners.

VII. Acknowledgments

The presented work is part of the research project *KRITIS Scalable Safe and Secure Modules* (KRITIS³M), which is funded by the Project Management Jülich (PtJ) and the German Federal Ministry for Economic Affairs and Climate Action (BMWK) under funding code 03EI6089A. Special thanks to Tobias Frauenschläger, Franziska Rasp and Peter Heller for their constructive feedback during the development of this paper.

References

[1] Council of the European Union. *Directive (EU) 2022/2555 of the European Parliament and of the Council of 14 December 2022 on measures for a high common level of cybersecurity across the Union, amending Regulation (EU) No 910/2014 and Directive (EU) 2018/1972, and repealing Directive (EU) 2016/1148 (NIS 2 Directive)*. L 333, p. 80. Dec. 2022.

[2] S. Dubroca. *ip-macsec(8) – Linux manual page*. https://man7.org/linux/man-pages/man8/ip-macsec.8.html. Accessed: 2024-05-13.

[3] Federal Office for Information Security (BSI). *BSI-Standard 200-1 Information Security Management Systems (ISMS)*. Version 1.0. Oct. 2017.

[4] S. M. Suhail Hussain, S. M. Farooq, and T. S. Ustun. "A method for achieving confidentiality and integrity in IEC 61850 GOOSE messages." In: *IEEE transactions on Power Delivery* 35.5 (2020), pp. 2565–2567.

[5] S. M. Suhail Hussain, T. S. Ustun, and A. Kalam. "A review of IEC 62351 security mechanisms for IEC 61850 message exchanges." In: *IEEE Transactions on Industrial Informatics* 16.9 (2019), pp. 5643–5654.

[6] *IEC 61850:2023 SER Series – Communication networks and systems for power utility automation – ALL PARTS*. International Standard. Geneva, CH: International Electrotechnical Comission, 2023.

[7] *IEC 61850:2023 SER Series – Communication networks and systems for power utility automation – Part 6 Configuration description language for communication in power utility automation systems related to IEDs*. International Standard. Geneva, CH: International Electrotechnical Comission, 2010.

[8] *IEC 61850:2023 SER Series – Communication networks and systems for power utility automation – Part 7-1 Basic Communication Structure – Principles and Models*. International Standard. Geneva, CH: International Electrotechnical Comission, 2011.

[9] *IEC 61850:2023 SER Series – Communication networks and systems for power utility automation – Part 7-3 Basic Communication Structure – Common data classes*. International Standard. Geneva, CH: International Electrotechnical Comission, 2010.

[10] *IEC 61850:2023 SER Series – Communication networks and systems for power utility automation – Part 7-4 Basic Communication Structure – Compatible logical node classes and data object classes*. International Standard. Geneva, CH: International Electrotechnical Comission, 2010.

[11] *IEC 61850:2023 SER Series – Communication networks and systems for power utility automation – Part 8-1 Specific communication service mapping (SCSM) – Mappings to MMS (ISO 9506-1 and ISO 9506-2) and to ISO/IEC 8802-3*. International Standard. Geneva, CH: International Electrotechnical Comission, 2011.

[12] *IEC 61850:2023 SER Series – Communication networks and systems for power utility automation – Part 90-5: Use of IEC 61850 to transmit synchrophasor information according to IEEE C37.118*. International Standard. Geneva, CH: International Electrotechnical Comission, 2012.

[13] *IEC 62351:2024 SER Series – Power systems management and associated information exchange - Data and communications security - ALL PARTS*. International Standard. Geneva, CH: International Electrotechnical Comission, 2024.

[14] *IEEE Std 802.1AE-2018 – IEEE Standard for Local and metropolitan area networks – Media Access Control (MAC) Security*. International Standard. New York, USA: The Institue of Electrical and Electronics Engineers, Inc., 2018.

[15] J. W. Konka et al. "Traffic generation of IEC 61850 sampled values." In: *2011 IEEE First International Workshop on Smart Grid Modeling and Simulation (SGMS)*. 2011, pp. 43–48. DOI: 10.1109/SGMS.2011.6089025.

[16] N. S. Kush et al. "Poisoned GOOSE: Exploiting the GOOSE protocol." In: *Proceedings of the Twelfth Australasian Information Security Conference (AISC 2014)[Conferences in Research and Practice in Information Technology, Volume 149]*. Australian Computer Society. 2014, pp. 17–22.

[17] T. Lackorzynski et al. "Enabling and optimizing MACsec for industrial environments." In: *IEEE Transactions on Industrial Informatics* 17.11 (2020), pp. 7599–7606.

[18] R. E. Mackiewicz. "Overview of IEC 61850 and Benefits." In: *2006 IEEE Power Engineering Society General Meeting*. IEEE. 2006, 8–pp.

[19] N. Moreira et al. "Cyber-security in substation automation systems." In: *Renewable and Sustainable Energy Reviews* 54 (2016), pp. 1552–1562.

[20] *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. Public Recommendation. Gaithersburg, USA: National Institute of Standards and Technology, 2007.

[21] M. Rodríguez et al. "A fixed-latency architecture to secure GOOSE and sampled value messages in substation systems." In: *IEEE Access* 9 (2021), pp. 51646–51658.

[22] SGRWin - Network Solutions Suite. *Basic understanding of IEC 61850*. https://www.sgrwin.com/goose-mms-and-sv-protocols/. Accessed: 2024-04-30.

[23] Typhoon HIL Inc. *IEC 61850 Sampled Values protocol*. https://www.typhoon-hil.com/documentation/typhoon-hil-software-manual/References/iec_61850_sampled_values_protocol.html. Accessed: 2024-04-30.

[24] T. S. Ustun, S. M. Farooq, and S. M. Suhail Hussain. "Implementing secure routable GOOSE and SV messages based on IEC 61850-90-5." In: *IEEE Access* 8 (2020), pp. 26162–26171.

[25] T. S. Ustun and S. M. Suhail Hussain. "IEC 62351-4 security implementations for IEC 61850 MMS messages." In: *IEEE Access* 8 (2020), pp. 123979–123985.

[26] "Zweites Gesetz zur Erhöhung der Sicherheit informationstechnischer Systeme*." In: *Bundesgesetzblatt, ausgegeben zu Bonn* (2021). Teil I Nr. 25, pp. 1122–1138.

Counter power leakage for frequency extraction of ring oscillators in ROPUF

Ondřej Staníček, Filip Kodýtek, Róbert Lórencz
Department of Information Security
Czech Technical University
Prague, Czech Republic
staniond@fit.cvut.cz, kodytfil@fit.cvut.cz, lorencz@fit.cvut.cz

Abstract

This paper deals with power side-channel analysis to extract frequencies of ring oscillators (ROs) to enable an attack on a ring oscillator based physical unclonable function (ROPUF). Side-channel attacks against ROPUFs exist, though they require costly equipment and are difficult to carry out. In this paper, we devise a method that uses a power side-channel to extract RO frequencies through counter leakage using less resources. This method allows to derive the PUF response of some ROPUF constructions, thus defeating them. We show the side-channel leakage on a minimal design consisting of a single RO and a counter. Then we explore the dependence of the leakage on FPGA routing and counter type and demonstrate the attack method on a ROPUF implemented on a Xilinx Artix-7 FPGA.

***Keywords*— PUF, ring oscillator, FPGA, side-channel attack, counter power leakage**

Paper Origin

Accepted for publication at the DSD 2024 conference.

An Evaluation of Hardware Accelerated MACsec in IEC 61850 compliant GOOSE Messaging

Sebastian Lintl, Florian Aschauer, Jürgen Mottok
Ostbayerisch Technische Hochschule (OTH)
Labratory for Safe and Secure Systems (LaS³)
Germany, Bayern, Regensburg 93053
Email: {sebastian.lintl, florian.aschauer, juergen.mottok} @oth-regensburg.de

Abstract

As critical infrastructure becomes increasingly interconnected, the need for efficient communication security to protect against vulnerabilities is growing. This paper investigates the protection of critical infrastructure, focusing on securing the communication of energy suppliers, as part of the KRITIS³M research project. The IEC 61850 standard defines protocols for communication among Intelligent Electronic Devices and demands real-time capability. A significant issue with the security demands of the standard is its lack of confidentiality. The IEC 62351 standard supplements GOOSE messages with integrity and authenticity mechanisms, but it does not cover confidentiality. In addition to this, the currently proposed security mechanisms present challenges for real-time capabilities due to long computation times. To address these security gaps, this paper proposes enhancing the IEC 61850 standard with MACsec. It outlines the implementation of MACsec for securing IEC 61850 and evaluates its effectiveness in meeting real-time requirements. It also evaluates performance optimization for real-time requirements using hardware-accelerated encryption with FPGA. A mathematical analysis proofs that MACsec can achieve the necessary transfer time of less than 3 ms for GOOSE and SV messages. Overall, the paper concludes that MACsec is a viable solution for ensuring integrity, authentication, and confidentiality in the communication of critical infrastructure.

***Keywords*— IEEE 802.1AE, MACsec, IEC61850, GOOSE, hardware acceleration, FPGA**

I. Introduction

The KRITIS³M research project aims to protect critical infrastructure, with a particular focus on the security of power grid. As critical infrastructure becomes more interconnected, the need to secure communications and prevent vulnerabilities continues to grow. The IEC 61850 standard defines protocols for the efficient communication of Intelligent Electronic Devices (IED)s and demands real-time capability. However, it does not specify security mechanisms. The IEC 62351 standard addresses the security gap in Generic Object Oriented Substation Events (GOOSE) messages by providing mechanisms focused on integrity and authenticity. However, it does not cover confidentiality, and some recommended mechanisms struggle with real-time capabilities due to lengthy computation times [1]. This paper introduces a promising alternative security implementation for IEC 61850 compliant communication using Media-Access-Control security (MACsec). To meet real-time requirements, the Authenticated Encryption with Associated Data (AEAD) cipher suite of the MACsec protocol is offloaded to an FPGA for hardware acceleration. Chapter II offers a basic overview of the standards IEC61850, 62351 IEEE 802.1AE and 802.1X, explaining the relevant frames in detail. Chapter III outlines existing concepts for protecting the IEC 61850 and how to apply them. Chapter IV details the integration of MACsec to secure IEC 61850. Chapter V evaluates the effectiveness of MACsec in the context of IEC 61850, including a theoretical time analysis to confirm real-time capability. Finally, Chapter VI summarizes the findings and provides an outlook on future research work.

II. Background

A. Overview of IEC 61850 Standard

The IEC 61850 gives protocols for time-critical communication. For secure real-time communication, the IEC 62351 standard provides the necessary mechanisms for the IEC 61850. The IEC 61850 standard facilitates communication within substations and across domains. It follows the Open Systems Interconnection (OSI) 7-layer model, ensuring flexibility and long-term stability. Within this framework, as shown in Figure 1, substation data services and applications operate above the application layer. [1]

The Abstract Communication Service Interface (ACSI) provides a standardized interface for IEDs, abstracting the specifics of the communication stack. The Specific Communication Service Mapping (SCSM) defines how ACSI services and objects are mapped onto specific protocols. Three communication models are specified for ACSI services: client/server communication, publisher/subscriber model with GOOSE and multicast Ethernet-based Sample Values (SV) messages [1]. Client/server communication, using Manufacturing Message Specification (MMS) over Transmission Control Protocol/Internet Protocol (TCP/IP), is widely used in IEC 61850 communications, ensuring reliable data transfer [1]. GOOSE and SV messages are services for time-critical communication, such as error/event recording, where protection function messages are transmitted directly via multicast at the OSI data link layer. GOOSE messages typically carry binary data such as indications, alarms, and trip signals, while SVs transmit raw data from current/voltage transformers to IEDs. [1]

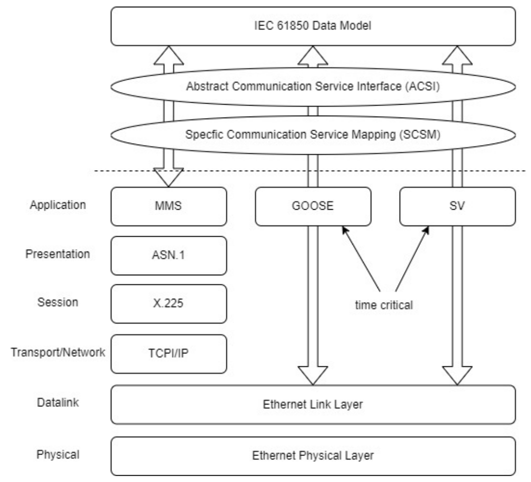


Fig. 1. IEC 61850 within the OSI-model [2]

B. *IEC 61850 GOOSE Frame*

The GOOSE frame, illustrated in Figure 2, comprises a `source` and `destination address`. In the Goose frame, the last two bytes of the destination address determine whether the message is relevant for the IED or can be discarded [3]. The IED utilizes its internal XML data structure to verify the relevance of the destination address. Furthermore, the frame contains a `VLAN-TAG`, the `GOOSE EtherType`, which is a hexadecimal code (0x88B8) identifying the frame as a GOOSE frame, the `AppID` (application identifier) is used to identify ISO/IEC 8802-3 frames containing GOOSE messages and differentiate between application associations. The `Length` field containing the total number of bytes of the Data Frame, AppID, `Reserved`, and GOOSE Protocol Data Unit (GPDU). The `frame check sequence (FCS)` is a checksum for error detection, and the GPDU contains the following fields [4]:

- `GoCBRef`: GOOSE Control Block Reference, providing the name of the GOOSE control block.
- `TTL`: Time to Live, indicating the maximum time a packet remains valid after transmission.
- `GoID`: GoID, buffer id [5]
- `timestamp`: Indicates the time at which the GOOSE message was generated.
- `state Num`: Assigned whenever a GOOSE message is generated due to an event change.
- `seq Num`: Assigned sequentially to retransmitted GOOSE messages.
- `Data`: Contains the information transmitted by the GOOSE message.

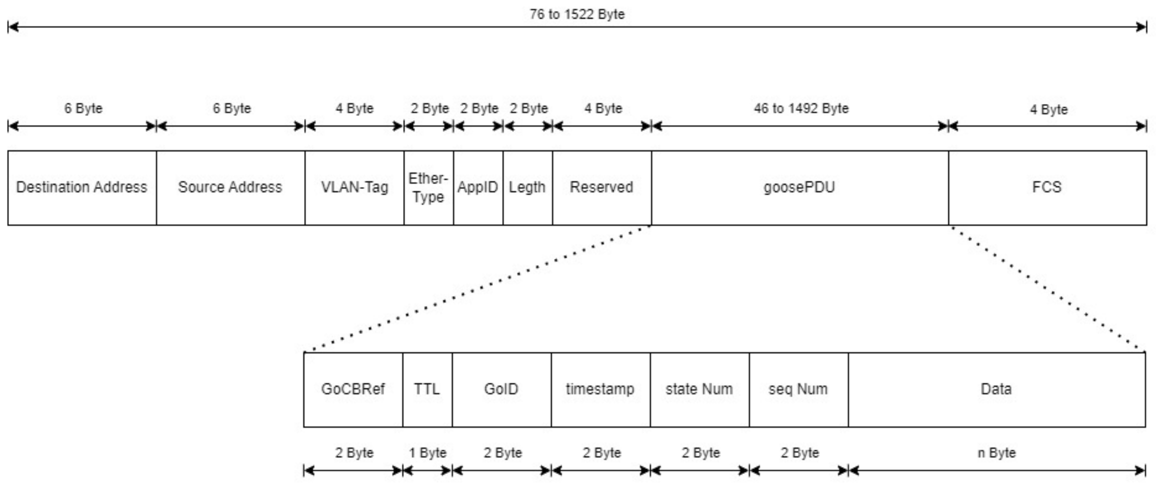


Fig. 2. GOOSE Frame [2]

C. Security of the IEC 61850-Standard

Incorporating cryptographic algorithms into IEDs faces a challenge due to constraints in memory and processing capabilities. The IEC 62351 specifies the security mechanisms of the IEC 61850. It recommends prioritizing authentication for messages relying on GOOSE or SV, which require strict real-time transmission, as shown in Table I. Therefore, mechanisms are used to ensure integrity and authenticity within real-time requirements, but they do not secure confidentiality. [1]

TABLE I
IEC61850 REAL-TIME REQUIREMENTS [2]

Application	Message Type	Time Requirements
Fast Messages	GOOSE	$\leq 3\,\mathrm{ms}$
Raw Data	SV	$\leq 3\,\mathrm{ms}$
Medium Speed Messages	MMS	$\leq 100\,\mathrm{ms}$
Low Speed Messages	MMS	$\leq 500\,\mathrm{ms}$
File Transfer	MMS	$> 500\,\mathrm{ms}$

The standard suggests securing GOOSE and SV messages with Message Authentication Codes (MAC) using Secure Hash Algorithm (SHA), which are digitally signed with the Rivest, Shamir and Adleman (RSA) public key cryptosystem to ensure source authenticity [1]. The primary drawback of RSA signatures is the lengthy execution times for both signature calculation and verification. Even with the use of a high-performance ARM processor featuring a crypto accelerator core in substation equipment, computing and verifying the RSA signature with 1024-bit keys within the 3 ms maximum transmission time required by GOOSE messages is impractical [1]. To mitigate the impact of security mechanisms on field device performance and fulfill all protection objectives, symmetric cryptography is proposed as an alternative to digital signatures. To meet the real-time demands and ensure encryption and confidentiality, the setup employs a fast logical path and cut-through switches. Symmetric cryptographic units are used only within end nodes, ensuring protection through end-to-end security mechanisms such as MACsec. This ensures that integrity, authentication, and confidentiality are maintained. [1]

D. MACsec Security Standard Overview

The IEEE 802.1AE standard ensures secure communication within hop-by-hop Ethernet connections. MACsec, operating at layer 2 (Datalink layer) of the OSI reference model, ensures the confidentiality and integrity of connections. With cut-through switches, MACsec enables secure communication point-to-point. For instance, with IEEE 802.1AE, layer-2 Local Area Network (LAN) connections between end devices and switches or between switches and routers can be encrypted and secured using MACsec [6]. It uses an AEAD cipher, such as Advanced Encryption Standard-Galois Counter Mode (AES-GCM), to ensure the confidentiality and integrity of all network traffic. The MACsec frame format, along with the cipher suite specified in the standard, streamlines hardware implementations, enabling efficient processing with minimal latency and high bandwidth. Consequently, the Time-Sensitive Networking (TSN) community regards MACsec as a promising security solution for TSN, particularly suited for securing GOOSE messages [7]. Within the MACsec Key Agreement (MKA), the Connectivity Association Key is utilized to generate transient session keys known as Secure Association Keys (SAK). These SAK, alongside other crucial control information, are disseminated in MKA protocol control packets. MACsec sessions are established exclusively, with a unique Secure Association Identifier (SAI) assigned to each session. The secure association is identified by an SAI, which is formed by combining the Secure Channel Identifier (SCI) with an Association Number (AN). [7]

E. MACsec Security Frame

The MACsec frame format, illustrated in Figure 3, comprises a `Security Tag (SecTAG)`, the `secured data`, and an `Integrity Check Value (ICV)`. The ICV provides a checksum for the entire MACsec frame. The security tag consists of the following components [7]:

- `MACsec EtherType`: This field is a 2-byte hexadecimal code (0x88e5) that identifies the following frame as a MACsec frame.
- `TCI`: Tag Control Information. This field enables version numbering, determination of whether confidentiality or integrity alone are in use, option inclusion, etc.
- `AN`: Association Number. It identifies up to four different secure associations within the context of a secure channel.
- `SL`: Short Length. This integer encodes the number of Bytes in the secure data field if the count is less than 48.
- `PN`: Packet Number. This field provides a unique initialization vector for all data transmitted using the same secure association, while also supporting replay protection.
- `SCI`: Secure Channel Identifier. Optionally encoded Secure Channel Identifier. This facilitates identification of the secure channel, particularly when dealing with three or more peers.

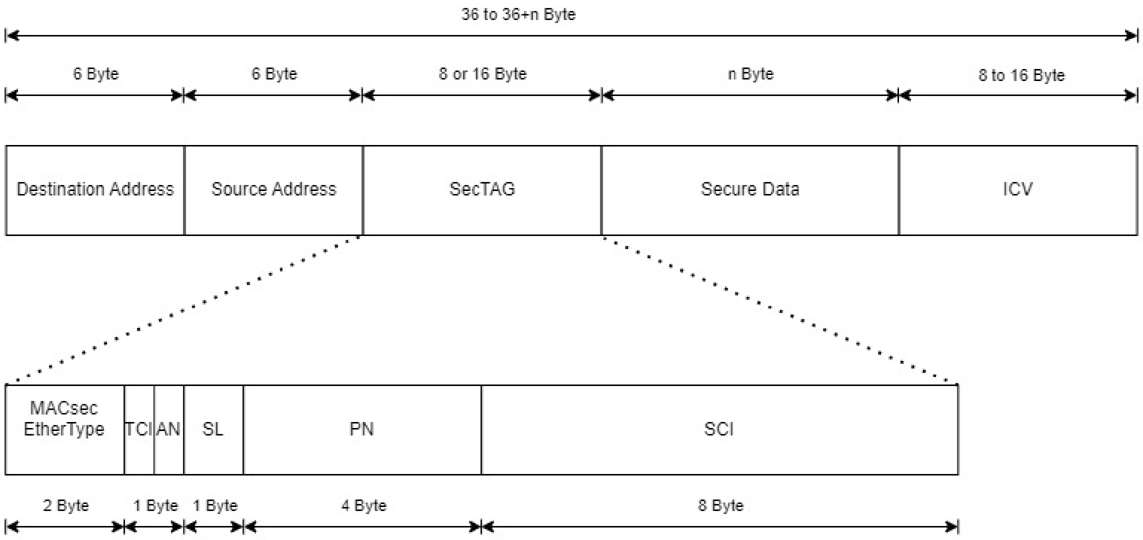


Fig. 3. MACsec Frame [6]

III. Related Work

This study builds upon the research of Moreira, Molina, Lázaro, *et al.* [1]. They present approaches for securing GOOSE. The investigation explores the standard outlined in IEC 62351, focusing on communication models and security mechanisms for present-day substations. However, safety deficiencies exist within these standards.

For instance, while the use of the RSA crypto system is mandated to guarantee the source authenticity of GOOSE and SV messages, its execution times exceed the maximum transfer times specified in the standard. This is problematic for time-critical applications, even when using costly processors with crypto accelerators execution times exceed the maximum transfer times. Additionally, the recommended time synchronization solution, the Precision Time Protocol (PTP) outlined in IEEE 1588, introduces an optional security extension based on keyed hash algorithms, which is suboptimal due to latency times and resource requirements [1]. The primary aim of this research is to check existing security solutions and assess their suitability for substation environments. Furthermore, as part of a proposed future security framework, the authors recommend a security approach based on MACsec. This approach enables the coexistence of various communication services with diverse performance and security requirements within the substation network. The evaluation of this paper concentrates on a fast logical path with cut-through switches and cryptographic units solely within end nodes for time-critical messages secured by point-to-point security mechanisms [1]. Additionally, encryption is planned to be accelerated with an Field Programmable Gate Array (FPGA) to ensure time-sensitive communication. [1]

Another significant paper influencing this research is from Rodriguez, Lazaro, Bidarte, *et al.*[2]. The IEC 62351 standard outlines security measures for securing real-time communications within the framework of IEC 61850. However, the stringent requirement of generating, transmitting, and processing GOOSE and SV messages within a 3 ms timeframe poses a challenge to its implementation. In response to identified security threats of the IEC 61850 communications and considering the current state of GOOSE and SV security, this study proposes an architecture centered on wire-speed processing, providing both message authentication and confidentiality. Implementation of this architecture was conducted, followed by performance evaluation, resource utilization assessment, and latency analysis. The FPGA architecture achieves authentication and encryption of real-time IEC 61850 data in under 7 µs, ensuring predictable latency, while also complying with IEC 62351:2020 requirements. [2]

Another paper [8] examines message authentication codes such as AES-GMAC or HMAC to secure the IEC 61850 messages. The authors Hussain, Farooq, and Ustun focus on meeting timing requirements for these message authentication codes but do not address confidentiality concerns [8].

This research aims to achieve outcomes similar to those in the paper [2], focusing on providing integrity, authentication, and confidentiality for GOOSE and SV messages. While the referenced paper proposes using a substandard of IEC 61850, this study seeks to evaluate the architecture using the MACsec standard.

IV. Implementation

To ensure the security of GOOSE and SV messages, this approach uses MACsec. Through MACsec, integrity, authentication, and confidentiality are assured. As both the MACsec security implementation and GOOSE messaging operate on the same level of the OSI model, the MACsec secured frame wraps the GOOSE frame. Switches classify the composite frame as a MACsec frame based on the first Ethertype it contains. Since MACsec encapsulates GOOSE, the first Ethertype in the frame is that of MACsec. The GOOSE package itself gets encrypted and can only be deciphered by the communication partner holding a valid SAK (Figure 4).

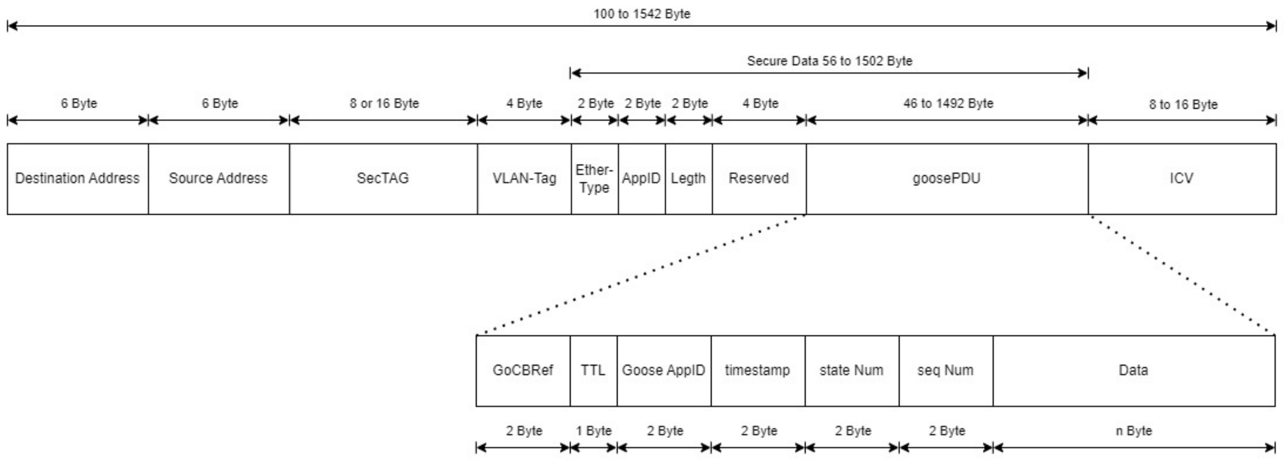
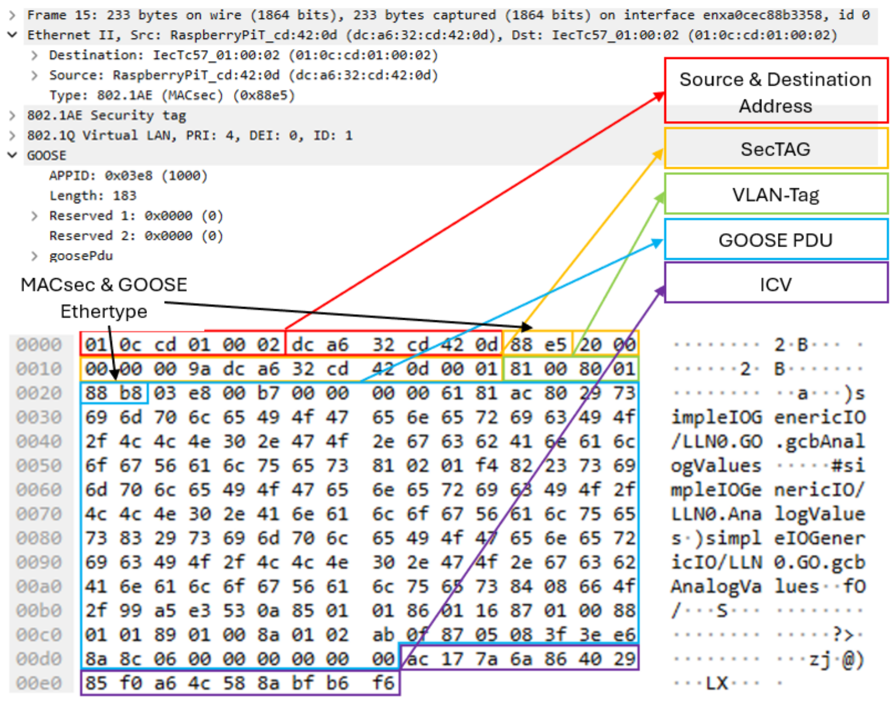


Fig. 4. Combined MACsec and Goose Frame

The validated composition of the MACsec-GOOSE frame is shown in Figure 5. This composite frame was validated using a test setup with two Raspberry Pi devices. The first Raspberry Pi creates the frame and sends the unencrypted frame over Ethernet to the second Raspberry Pi.



```
> Frame 15: 233 bytes on wire (1864 bits), 233 bytes captured (1864 bits) on interface enxa0cec88b3358, id 0
v Ethernet II, Src: RaspberryPiT_cd:42:0d (dc:a6:32:cd:42:0d), Dst: IecTc57_01:00:02 (01:0c:cd:01:00:02)
   > Destination: IecTc57_01:00:02 (01:0c:cd:01:00:02)
   > Source: RaspberryPiT_cd:42:0d (dc:a6:32:cd:42:0d)
     Type: 802.1AE (MACsec) (0x88e5)
> 802.1AE Security tag
> 802.1Q Virtual LAN, PRI: 4, DEI: 0, ID: 1
v GOOSE
     APPID: 0x03e8 (1000)
     Length: 183
   > Reserved 1: 0x0000 (0)
     Reserved 2: 0x0000 (0)
   > goosePdu

0000  01 0c cd 01 00 02 dc a6  32 cd 42 0d 88 e5 20 00   ········ 2·B··· ·
0010  00 00 00 9a dc a6 32 cd  42 0d 00 01 81 00 80 01   ······2· B·······
0020  88 b8 03 e8 00 b7 00 00  00 00 61 81 ac 80 29 73   ········ ··a···)s
0030  69 6d 70 6c 65 49 4f 47  65 6e 65 72 69 63 49 4f   impleIOG enericIO
0040  2f 4c 4c 4e 30 2e 47 4f  2e 67 63 62 41 6e 61 6c   /LLN0.GO .gcbAnal
0050  6f 67 56 61 6c 75 65 73  81 02 01 f4 82 23 73 69   ogValues ·····#si
0060  6d 70 6c 65 49 4f 47 65  6e 65 72 69 63 49 4f 2f   mpleIOGe nericIO/
0070  4c 4c 4e 30 2e 41 6e 61  6c 6f 67 56 61 6c 75 65   LLN0.Ana logValue
0080  73 83 29 73 69 6d 70 6c  65 49 4f 47 65 6e 65 72   s·)simpl eIOGener
0090  69 63 49 4f 2f 4c 4c 4e  30 2e 47 4f 2e 67 63 62   icIO/LLN 0.GO.gcb
00a0  41 6e 61 6c 6f 67 56 61  6c 75 65 73 84 08 66 4f   AnalogVa lues··fO
00b0  2f 99 a5 e3 53 0a 85 01  01 86 01 16 87 01 00 88   /···S··· ········
00c0  01 01 89 01 00 8a 01 02  ab 0f 87 05 08 3f 3e e6   ········ ·····?>·
00d0  8a 8c 06 00 00 00 00 00  00 ac 17 7a 6a 86 40 29   ········ ···zj·@)
00e0  85 f0 a6 4c 58 8a bf b6  f6                        ···LX··· ·
```

Fig. 5. Combined unencrypted MACsec Goose Frame track

The sequence of frame assembly and processing is partially outlined in IEEE 802.1AE [6]. To simulate and evaluate the secure communication, two prototypes are planned. The test setup utilizes Raspberry Pi 5 devices to simulate the IED and FPGAs configured for acceleration (Figure 6) to evaluate secure communication. Initially, for the first prototype, the Raspberry Pi generates a hybrid MACsec-GOOSE frame that includes parts of the message but not the entire MACsec-GOOSE frame. The FPGA also includes a processor capable of generating the hybrid MACsec-GOOSE frame. Initially, the setup uses a kernel module on the Raspberry Pi for easier prototyping and validation. In a further prototype, the FPGA will generate the entire MACsec-GOOSE frame, with the Raspberry Pi 5 serving only as a test IED, sending unencrypted data to the FPGA. In the first setup, the missing parts of the MACsec-GOOSE frame is calculated by the FPGA. The assembled frame is then transmitted via Peripheral Component Interconnect Express (PCI-E) to the FPGA in accelerator configuration. PCI-E is used for its fast communication link and the potential to integrate it into future prototypes for communicating with IEDs, such as server racks. Within the FPGA, the Secure-Data encryption and ICV calculation occur simultaneously, utilizing AES-GCM-128 encryption. After encryption, the FPGA returns the combined frame to the Raspberry Pi, which proceeds to transmit the encrypted message. A second Raspberry Pi forwards the encrypted MACsec-GOOSE-Frame to the FPGA for decryption. After decryption, the FPGA sends back the message to the Raspberry Pi. When the Raspberry Pi receives the encrypted message, the message is delivered.

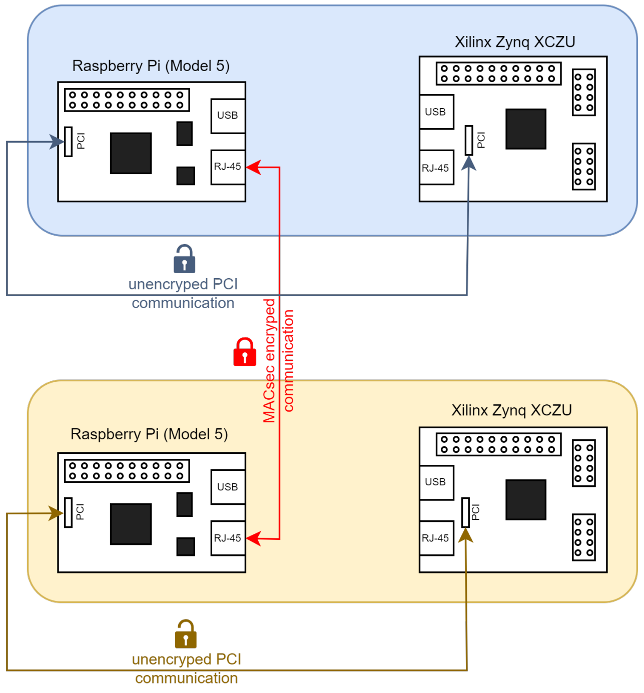


Fig. 6. Hardware architecture for the MACsec-Goose communication

V. Evaluation

MACsec and GOOSE are two protocols that are already used in the industry. Both are reliable and validated. What has not yet been validated is the combination of the two protocols. Therefore, the evaluation will focus on this aspect. The evaluation primarily focuses on the analysis of time. As described in Table I, both GOOSE and SV messages are not allowed to take longer than 3 ms.

The following calculations describe the transfer times for data over a PCIe link under specific conditions. PCI_{link} represents the PCIe speed, and $PCI_{\text{Transfer-Size}}$ indicates the amount of data that can be transmitted at once. n_{Byte} is the total size of the frame to be transmitted. The number of transfers is calculated by dividing the total bytes to be sent by the data per transfer.

- The Transaction Layer Packets (TLP) to Acknowledgment (ACK) ratio is 1 ACK for every 5 TLPs.
- An Flow control (FC) update is issued every 4 TLPs.
- $PCI_{\text{link}} = 5.0\,\text{GB}\,\text{s}^{-1}$= 1 byte every 2ns
- $PCI_{\text{Transfer-Size}} = 128\,\text{Byte}$
- $n_{\text{Byte}} = 100$ to $1542\,\text{Byte}$

- $n_{\text{TLPmax}} = \left\lceil \frac{1542\,\text{bit}}{128\,\text{bit}} \right\rceil = 13$
- $n_{\text{TLPmin}} = \left\lceil \frac{100\,\text{bit}}{128\,\text{bit}} \right\rceil = 1$

The PCI transfer time calculation is based on the example 1 provided by Xilinx [9]. The PCI overhead is 20 Bytes. Depending on the lane width of the PCIe, t_{transfer} is divided by the Bytes per lane to be transmitted. The amount of $t_{\text{ACK/TLP}}$ transmission depends on the specifications given above.

$$t_{\text{transfer}} = \frac{128\,\text{Byte} + \mathit{overhead} = 20\,\text{Byte}}{1\,\text{Byte}} \cdot 2\,\text{ns} = 296\,\text{ns} \tag{1}$$

$$t_{\text{ACK/TLP}} = \frac{8\,\text{Byte}}{1\,\text{Byte}} \cdot 2\,\text{ns} = 16\,\text{ns} \tag{2}$$

$$n_{\text{ACK/100Byte}} = \left\lceil \frac{n_{\text{TLPmin}} = 1}{n_{\text{ACK}} = 5} \right\rceil = 1 \tag{3}$$

$$n_{\text{ACK/1546Byte}} = \left\lceil \frac{n_{\text{TLPmax}} = 13}{n_{\text{ACK}} = 5} \right\rceil = 3 \tag{4}$$

$$n_{\text{FC/100Byte}} = \left\lceil \frac{n_{\text{TLPmin}} = 1}{n_{\text{FC}} = 4} \right\rceil = 1 \tag{5}$$

$$n_{\text{FC/1546Byte}} = \left\lceil \frac{n_{\text{TLPmax}} = 13}{n_{\text{FC}} = 4} \right\rceil = 4 \tag{6}$$

$$t_{100\,\text{Byte}} = 1 \cdot 296\,\text{ns} + 1 \cdot 16\,\text{ns} + 1 \cdot 16\,\text{ns} = 328\,\text{ns} \tag{7}$$

$$t_{1546\,\text{Byte}} = 13 \cdot 296\,\text{ns} + 4 \cdot 16\,\text{ns} + 3 \cdot 16\,\text{ns} = 3690\,\text{ns} \tag{8}$$

The time t_{encrypt} represents the total encryption duration. The conditions for the encryption are given below. f_{clk} is the FPGA frequency, and n_{Byte} is the total number of Bytes.

- $f_{\text{clk}} = 100\,\text{MHz}$
- $n_{\text{Byte}} = 56$ to $1502\,\text{Byte}$

The calculation is based on the encryption timing formula for AES-128 [10]. The AES core encrypts 128 bits per clock cycle.

$$t_{\text{encrypt}} = \left\lceil \frac{n_{\text{Byte}} \cdot 8}{128\,\text{bit}} \right\rceil \cdot \frac{1}{f_{\text{clk}}} = 40\,\text{ns} - 940\,\text{ns} \tag{9}$$

Table II lists the individual steps and the time they need. The total result of the secure communication is, in the worst case, 24 531 ns. The total time significantly falls below the maximum allowed communication time.

TABLE II
TIMING OF MACSEC-GOOSE-FRAME

Number	Step	Source	Equation	Time
1	Create MACsec/Goose frame	[11]		300 ns
2	Transfer MACsec/Goose frame to FPGA via PCI		Equation 7, 8 based on [9]	328 ns-3960 ns
3	Encrypt Goose frame and calculate ICV		Equation 9 based on [10]	40 ns-940 ns
4	Transfer encrypted MACsec/Goose frame to PI via PCI		Equation 7, 8 based on [9]	328 ns-3960 ns
5	Transfer MACsec-Goose-Frame (Cut-Through switching mode)	[7]		6511 ns
6	Transfer encrypted MACsec/Goose frame to FPGA via PCI		Equation 7, 8 based on [9]	328 ns-3960 ns
7	Decrypt Goose frame and calculate ICV		Equation 9 based on [10]	40 ns-940 ns
8	Transfer MACsec/Goose frame to PI via PCI		Equation 7, 8 based on [9]	328 ns-3960 ns
total				8203 ns-24 531 ns

This calculation confirms that the communication system shown in Figure 6 meets the required speed of less than 3 ms. This setup is optimized for communication between two clients. However, paper [12] indicates that additional clients on the same communication path can significantly reduce communication speed, potentially doubling the required time. For clarification, it is necessary to evaluate the performance in a real scenario.

VI. Conclusion

The paper examines the security of GOOSE and SV messages. The IEC 61850 standard suggests security mechanisms that only ensure the integrity and authentication of the data. However, confidentiality is not protected by these mechanisms. To ensure security even for low-performance IEDs, MACsec with a symmetric encryption algorithm (AES-GCM-128) is investigated. MACsec can ensure integrity, authenticity, and confidentiality with minimal computational overhead. The efficiency of MACsec is advantageous as GOOSE communication requires a maximum transfer time of less than $3\,\mathrm{ms}$. The practicability of the GOOSE-MACsec-Frame was examined and confirmed through a test setup. Furthermore, the expected communication duration is evaluated mathematically. The communication duration is under 3 ms, therefore MACsec can be used for GOOSE and SV messages to protect integrity, authentication, and additionally confidentiality. After the successful examination, the next step is to implement and test the setup to compare the real-world results with the calculated values and demonstrate its potential.

Acknowledgement

The presented work is part of the research project *KRITIS Scalable Safe and Secure Modules* (KRITIS³M), which is funded by the Project Management Jülich (PtJ) and the German Federal Ministry for Economic Affairs and Climate Action (BMWK) under funding code 03EI6089A. Special thanks to Lukas Fuereder, Tobias Frauenschläger, and Franziska Rasp for their reviews.

References

[1] N. Moreira, E. Molina, J. Lázaro, E. Jacob, and A. Astarloa, “Cyber-security in substation automation systems,” *Renewable and Sustainable Energy Reviews*, vol. 54, pp. 1552–1562, Feb. 1, 2016, ISSN: 1364-0321. DOI: 10.1016/j.rser.2015.10.124. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364032115012034 (visited on 05/06/2024).

[2] M. Rodriguez, J. Lazaro, U. Bidarte, J. Jimenez, and A. Astarloa, “A fixed-latency architecture to secure GOOSE and sampled value messages in substation systems,” *IEEE Access*, vol. 9, pp. 51 646–51 658, 2021, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3069088. [Online]. Available: https://ieeexplore.ieee.org/document/9387365/ (visited on 05/06/2024).

[3] “Nautos,” [Online]. Available: https://nautos.de/TQR/search/item-detail/DE30093337 (visited on 05/23/2024).

[4] C. Fernandes, S. Borkar, and J. Gohil, “Testing of goose protocol of IEC61850 standard in protection IED,” *International Journal of Computer Applications*, vol. 93, no. 16, pp. 30–35, May 16, 2014, ISSN: 09758887. DOI: 10.5120/16301-6112. [Online]. Available: http://research.ijcaonline.org/volume93/number16/pxc3896112.pdf (visited on 05/13/2024).

[5] T. S. Ustun, S. M. Farooq, and S. M. S. Hussain, “A novel approach for mitigation of replay and masquerade attacks in smartgrids using IEC 61850 standard,” *IEEE Access*, vol. 7, pp. 156 044–156 053, 2019, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2948117. [Online]. Available: https://ieeexplore.ieee.org/document/8873588 (visited on 05/30/2024).

[6] “IEEE std 802.1ae™-2018, IEEE standard for local and metropolitan area networks—media access control (MAC) security,” 2018.

[7] R. A. Peña, M. Pascual, A. Astarloa, D. Uribe, and J. Inchausti, “Impact of MACsec security on TSN traffic,” in *2022 37th Conference on Design of Circuits and Integrated Circuits (DCIS)*, ISSN: 2640-5563, Nov. 2022, pp. 01–06. DOI: 10.1109/DCIS55711.2022.9970155. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9970155 (visited on 05/13/2024).

[8] S. M. S. Hussain, S. M. Farooq, and T. S. Ustun, “Analysis and implementation of message authentication code (MAC) algorithms for GOOSE message security,” *IEEE Access*, vol. 7, pp. 80 980–80 984, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2923728. [Online]. Available: https://ieeexplore.ieee.org/document/8740995/ (visited on 05/14/2024).

[9] J. Lawley, “Understanding performance of PCI express systems,” 2014. [Online]. Available: https://docs.amd.com/v/u/en-US/wp350.

[10] Luca, *BLu85/AES-GCM-128-192-256-bits*, original-date: 2021-04-18T21:13:03Z, May 5, 2024. [Online]. Available: https://github.com/BLu85/AES-GCM-128-192-256-bits (visited on 05/07/2024).

[11] D. Dik, I. Larsen, and M. Stübert Berger, “MACsec and AES-GCM hardware architecture with frame preemption support for transport security in time sensitive networking,” in *2023 International Conference on Computer, Information and Telecommunication Systems (CITS)*, Genoa, Italy: IEEE, Jul. 10, 2023, pp. 01–07, ISBN: 9798350336092. DOI: 10.1109/CITS58301.2023.10188711. [Online]. Available: https://ieeexplore.ieee.org/document/10188711/ (visited on 05/24/2024).

[12] S. Secci, G. Pujolle, T. M. T. Nguyen, and S. C. Nguyen, “Performance–cost trade-off strategic evaluation of multipath TCP communications,” *IEEE Transactions on Network and Service Management*, vol. 11, no. 2, pp. 250–263, Jun. 2014, Conference Name: IEEE Transactions on Network and Service Management, ISSN: 1932-4537. DOI: 10.1109/TNSM.2014.2321838. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6811210?casa_token=RdgJ9wn9qHwAAAAA:Jp-_U_K_NZOePLKjJVRGvgeiocfq7goACBDNnbS3Fr6iSHWoJJutg0-x_xSFI9h8GTlWhrg (visited on 06/24/2024).

Partners

Racyics

Racyics® is the leading System-on-Chip design partner in Europe. With Germany-wide locations in Dresden, Frankfurt/Main and Duisburg and counting over 120 employees, the company offers design services for analog, mixed-signal, and digital ICs including custom IP and turnkey ASIC solutions. Having worked for leading semiconductor companies for many years, the Racyics team has contributed to more than 100 successful chip designs down to 4nm feature size for automotive, consumer and communication applications. As GlobalFoundries® channel partner with focus on advanced and leading-edge technologies, Racyics provides access to 28nm, 22nm and 12nm prototyping runs (MPWs). Furthermore, Racyics offers with makeChip an unique design enablement service for Start-ups, SMEs and academia.

Tropic Square

Tropic Square is a fabless chip and IP company based in Prague that is at the forefront of open and transparent secure semiconductor innovation. The company stands for TRuly Open Integrated Circuits where the focus is on advancing hardware security through developing open source secure cryptographic coprocessors and IPs. The designs are tested by experts in the open source community and made available for community use. This approach enables embedded system designers to verify the security implementations while making informed decisions about their products' security frameworks and threat management strategies.

Sponsors

ASICentrum

ASICentrum, established in 1992 in Prague is a design center of EM Microelectronic and a competence center of ETA, belonging to the Swatch Group. EM Microelectronic is one of the most innovative IC providers. It developed and manufactured the smallest and the lowest power consuming Bluetooth chip on the market, the top performing optical sensors for optical office as well as gaming mice and it was the first to release the award-winning world-first dual-frequency NFC + RAIN RFID em|echo.

daiteq

daiteq, established in 2013, provides advanced arithmetic solutions for space-grade processors and FPGAs. For floating-point processing we offer a highly configurable floating-point units compliant with IEEE 754 (2019) that support complex arithmetic and user-defined floating-point number formats, beneficial e.g. for deep learning. For fixed-point processing we offer an implementation of SIMD operations targeted at GNSS processing, low-precision deep learning inference and video compression. Our arithmetic units are complemented by our customised version of the LLVM compiler that supports user-defined floating-point and integer data types.

IMA

IMA is a Czech ICT company specializing in the areas of identification, location detection, evidence and Internet of Things. IMA has a long-standing profile as an independent centre focused on the development and application of microcomputer electronics. In 2017, IMA started working with the German company WITTE Automotive on innovative gesture recognition systems. WITTE Automotive Group, of which IMA became a full partner on January 1, 2021, is a leader in the field of mechatronic locking systems and a major business group with a global presence. Our systems are used daily by hundreds of thousands of people worldwide by customers such as Škoda Auto, ČVUT, ČEZ, mBank, LEGO . . . We develop smart and innovative identification solutions and always strive to stay a few steps ahead of the competition. Participation in international grant projects aimed at finding new useful solutions for the future helps us to do this.

METIO Software

Metio Software is a software development company that develops various kinds of software projects.

STMicroelectronics

STMicroelectronics is a world leader in providing the semiconductor solutions that make a positive contribution to people's lives, today and into the future. ST is a global semiconductor company with net revenues of $17.3 billion in 2023. Offering one of the industry's broadest product portfolios, ST serves customers across the spectrum of electronics applications with innovative semiconductor solutions for Smart Driving and the Internet of Things. By getting more from technology to get more from life, ST stands for life.augmented.

SYSGO

SYSGO is the leading European provider of real-time operating systems for critical embedded applications. Our products have been designed to meet the highest requirements when it comes to Safety and Security. Our customers are leading players in the Avionics & Defense, Space, Railway, Automotive and Industrial Automation and Medical industries, who use our PikeOS product as a platform for critical systems that need to be certified against industry-specific Safety and Security standards.

UJP Praha

UJP PRAHA is a Czech company focused on research, development and manufacturing of globally distributed radiotherapy medical systems. Its research and development activities in the field of microelectronics focus on radiation-hardened microelectronics and the development of custom ASICs designed for extreme radiation operating conditions. It is currently the project leader of the Novel Radiation Tolerant Microelectronics for Medical, Scientific, Industrial, Space and Environmental applications which is supported by EU in IPCEI (Important Projects of Common European Interest).

IEEE Student Branch at Czech Technical University in Prague

IEEE Young Professionals

Computer (C) Society Chapter of the Czechoslovakia Section of IEEE

Partner conferences

29th IEEE European Test Symposium 2024