Using KNX-Based Building Automation and Control Systems for Data Exfiltration

Vitor Graveto, Tiago Cruz, *Senior Member, IEEE*, and Paulo Simões, *Senior Member, IEEE*

***Abstract*—When it comes to protecting confidential and/or sensitive information, organizations have a plethora of recommendations, standards, policies, and security controls at their disposal, conceived to deal with a wide variety of threats. However, most of them share the same fundamental premise: that weaknesses are inline by nature, as a consequence of infrastructure, social and/or technological gaps that can be detected, controlled, mitigated or constrained. Side channel threats are a different matter, though. Stemming from unconventional intrusion or attack vectors whose existence was inconceivable, unexpected or deemed unfeasible, their successful exploitation may provide attackers with the means to bypass and render most security controls ineffective or even useless. In this article, we address one such case: the use of a KNX-based building automation and control system to exfiltrate data from an air-gapped infrastructure. The introduction of a small device provides connectivity to the existing KNX fieldbus and enables sending data through it or even control other devices, with no interference in the operation of the building automation and control network. We validated the feasibility of this approach by means of an experimental setup, which was used to successfully evaluate two different techniques: 1) inline bus exfiltration and 2) optical transmission, via dimmer control. Finally, some measures for detecting and mitigating this type of attacks are proposed.**

***Index Terms*—Air gap, building automation and control system (BACS), BACSs, covert channel, exfiltration, KNX, safety, security.**

I. Introduction

ISOLATION is often regarded as a sound strategy to increase security, preserve sensitive data, and safeguard critical information. Thus, when an organisation needs to handle critical and sensitive information, it frequently resorts to physically isolated rooms and computers without any external network connection, as it is the case for Sensitive Compartmented Information Facilities or Special Access Program Facilities [1], [2]. Whenever confidential and highly sensitive information is present, the security of these air-gaped systems (i.e., systems without any Internet or even local network connection) can be increased using techniques, such as electromagnetic radiation isolation (through Faraday cages), frequency jammers (to prevent any wireless communications with the exterior), or acoustic insulation.

Quite often, organisations with those special security and confidentiality requirements are physically hosted in buildings equipped with automation systems, typically designated as building automation and control systems (BACSs). Historically, these systems base their security on infrastructure isolation and on the use of specific protocols, leading to several vulnerabilities, weaknesses and security gaps—as discussed in [3] and [4]. Due to their vital role in managing the operational needs, from lightning to climate control, BACS are capillary by nature, spreading over the entire building in an almost pervasive fashion.

This article demonstrates to which extent BACS can be leveraged to exfiltrate sensitive data outside secure/isolated spaces designed to protect confidential information or systems. This is achieved by using the BACS fieldbus network as a covert channel for the attacker to transmit information. Two ways of exfiltrating data were developed and will be presented: 1) the sending or streaming of data with potentially sensitive information (cf. Section IV-A) and 2) the usage of the lighting system for transmission of data in real time (cf. Section IV-B). These proof-of-concept attacks also challenge the perspective that operation technology (OT) network security, whether industrial or for building control, should be based on the implicit premise that the main threats often arise from interconnections with information technology (IT) assets, such as local networks or the Internet, and not within the BACS itself.

The remainder of this article is organised as follows. First, we review previous works on exfiltration in general (Section II). Next, we introduce KNX-based BACS (Section III), since this is the specific BACS variant we used in our proof-of-concept demonstration. Next, we present the proposed BACS exfiltration techniques (Section IV), along with a proof-of-concept implementation (Section V). Section VI addresses validation of the proposed techniques and possible countermeasures are discussed in Section VII. Finally, Section VIII concludes this article.

II. Review of Side Channel Exfiltration

The need to safeguard access to sensitive data and information has always existed and will most likely persist

Manuscript received 24 January 2023; accepted 25 March 2023. Date of publication 29 March 2023; date of current version 25 July 2023. This work was supported in part by the—Foundation for Science and Technology (FCT), I.P./MCTES through national funds (PIDDAC), within the scope of CISUC Research and Development Unit under Grant UIDB/00326/2020 or Project Code UIDP/00326/2020; and in part by the FEDER, in the context of the Competitiveness and Internationalization Operational Program (COMPETE 2020) of the Portugal 2020 Framework, in the scope of Project Smart5Grid under Grant POCI-01-0247-FEDER-047226. *(Corresponding author: Tiago Cruz.)*

The authors are with the University of Coimbra, CISUC, DEI, 3030-290 Coimbra, Portugal (e-mail: vgraveto@dei.uc.pt).

Digital Object Identifier 10.1109/JIOT.2023.3262873

in the future. To cope with this challenge, the research community has been long working on the identification of potential security gaps, in order to develop prevention, mitigation, and defense mechanisms against possible exploits.

One such case are exfiltration attempts, which allow a malicious actor to stealthily extract sensitive information from specific systems or places. In this context, the term *Bridgeware* was originally coined by Guri and Elovici [5] to designate malware that is designed to bridge the air gap between an isolated environment and an attacker. After breaching the isolated infrastructure, bridging is vital to make it possible to exfiltrate data, often by resorting to covert channels, which may also be useful to bypass existing protection mechanisms in noncontained environments. Covert channels identified and implemented in the past encompass several mediums, such as electromagnetic radiation, acoustic waves, thermal radiation, or optical transmission (e.g., LED, visible light, infrared) among others. This section presents a brief chronological review of several works that have been developed in this area.

Loughry and Umphress [6] reviewed existing knowledge and studied the potential for information leakage through optical LED emanation. This work complemented the scarce information available at the time about the topic, mostly available from Anderson's book [7] (whose first edition dates from 2001) and seminal paper [8]—the lack of publicly available information at the time was also due to the classification of all related information as sensitive by the U.S. National Security Agency.

Loughry and Umphress concluded that it is possible to read information from modulated optical signals that carry data injected by an eavesdropper into LEDs of various devices, such as modems, storage, routers, and other miscellaneous devices. They also demonstrated that it was possible to reconstruct error-free data at speeds of up to 56 kbit/s, being theoretically possible to go up to speeds of about 10 Mbit/s using the same fundamental principles. Moreover, a taxonomy for classifying optical emanations according to their risk level was proposed and used in tests performed on various market equipment, to assess their potential risk, with some possible countermeasures also being analyzed.

Hanspach and Goetz [9] developed a covert acoustic mesh using commonly available speakers and microphones operating in the ultrasonic frequency range. This mesh was based on an underwater communications stack, whose central frequency was changed from 4.2 to 21 KHz to guarantee its operation as a covert and stealthy communication channel. The use of multiple hops allows information to be sent over considerable distances—tests using conventional laptops (Lenovo T400) were successful with distances of up to 25 m. The authors also proposed some countermeasures, such as using an audio filter guard connected to the audio input and output devices, implementing a trusted bandpass or lowpass filter. This approach was also studied later by Carrara and Adams [10], extending it to several commercial equipment and analysing the relationship between distances and data transmission speeds, both in open spaces and closed environments, also resorting to error-correction algorithms. Some tests were also carried out at audible frequencies, thus, allowing the increase of transfer rates (which become available for overnight attacks) by widening the spectrum of usable frequencies.

Matyunin et al. [11] demonstrated the use of electromagnetic emissions emanating from a laptop, together with the ability for a mobile phone to sense magnetic fields, to create a covert channel for data exfiltration. Using this technique did not require any specific hardware. Writes between CPU and RAM, and from it to storage, have been found to produce electromagnetic fluctuations which can be modulated to convey information. The distance between devices has to be small (up to 4 cm) in order to allow for their successful use with a rate of up to 2 bits/s. Proposed countermeasures include shielding electronic laptop components, performing I/O operations randomly to mask electromagnetic emissions and, on the mobile phone side, to make it mandatory for applications to request permissions to access magnetic sensors (which has meanwhile been implemented in most recent mobile operating systems).

Robles-Durazno et al. [12] used two lamps connected to a programmable logic controller (PLC) for data exfiltration from an Industrial Automation and Control System. The inability for humans to detect flickering in the order of 60 Hz was taken into account, in order to avoid detection and make it a stealthy process. Transmitted information was received by a camera placed with line of sight to the transmitter, and then decoded.

Naz and Zeki [13] performed a review of various attack methods on air-gapped systems, summarizing used methods and devices, the attack effectiveness, and potential countermeasures.

In the last years the Ben-Gurion Univerity of the Negev has been producing a significant set of contributions in this field, developing several exfiltration techniques, such as those summarized below.

1) MOSQUITO [14] inverts the operation of the computer speakers, turning them into input devices. This technique is used to exchange ultrasonic waves with the purpose of creating a covert channel between two air-gapped computers, setting up a speaker-to-speaker link. As receivers, headphones, earphones, and earbuds were also tested, obtaining a transmission capacity between 300 and 600 bps at distances between 1 and 8 m. Even though some spaces do not allow audio recording as a security measure, this proved to be an insufficient restriction for the proposed technique, which leads to the need of alternative countermeasures, such as ultrasonic jammers. Overcoming countermeasures, such as removing speakers, by resorting to computer fans and other sound sources, was also researched [15].
2) CD-LEAK [16] proposes the usage of CD/DVD drives for the emission of acoustic signals in air-gapped computers without any audio equipment, for data exfiltration purposes. The analysis of acoustic signals emitted by the mechanical components of optical drives for CD, DVD or Blu-ray (tray loading mechanism, the motor used to open/close the tray, and the drive motor) allowed the creation of algorithms for the controlled emission of these acoustic signals. As a receiver, another computer

or mobile device with a network connection can be used, allowing a second hop in the exfiltration of information to the outside.

3) AirHopper [17] allows to exfiltrate data from air-gaped computers by leveraging the FM radio reception capability built into many mobile phones, without the owners' knowledge. Assuming that the attacker was able to previously deploy appropriate malware on the target computer, it becomes possible to emit radio signals by modulating the electromagnetic radiation generated by the video display adapter. These signals are then picked by a compromised employee's mobile phone (whose use is normally allowed) once it comes close to the compromised computer. A similar approach, based on radiation from USB cables, is proposed in [18].
4) BeatCoin [19] details how private keys could be leaked from an air-gapped computer where a cryptocurrency wallet is managed in offline mode. Several known techniques were explored, such as physical (removable media), electromagnetic, electric, magnetic, optical, and acoustic.
5) ODINI [20] allows the exfiltration of data from a computer that is both air gapped and placed in a Faraday-cage, exploiting the electromagnetic fields generated by the computer's CPU. Increasing the number of calculations to be performed by the CPU consumes more current and, consequently, generates a stronger magnetic field. The workload of each core is managed independently, which allows greater control over the generated magnetic field. Also, magnetic field control using virtual machines was explored with good results. Moreover, the Faraday-cage can be "bypassed" due to the fact that low frequency magnetic fields penetrate metals.
6) BitWhisper [21] makes it possible to establish a communication channel between two adjacent computers without any physical or radio frequency connection, using a covert channel that exploits the thermal radiation emitted by the devices. On infected computers, the BitWhisper malware uses the heat emanating from the CPU and other components (like the GPU, I/O controllers or mechanical systems, such as hard disks or optical drives), as well as the multiple temperature sensors that normally exist, to emit and receive heat, modulating commands over it to create a new covert channel. The handshaking process using thermal pings was also implemented in the developed prototype. Several tests were conducted, using a mix of distances, times, and temperatures. A similar cover channel implementation, using heating ventilation and air conditioning (HVAC) equipment, is proposed in [22].
7) Nassi et al. [23] showcased how to create a covert channel between a Command and Control (C&C) server and a malware installed in a computer by resorting to a document scanner and using it as a means of interaction.
8) The possibility of creating a covert channel using the activity LEDs of a compromised network-connected equipment is discussed in [24]. A compromised host on the network generates a blinking LED pattern by controlling the traffic flow on the corresponding LAN port, encoding the required binary data; if root/admin privileges are obtained on the compromised device, the host color of the status LED can also be controlled, by manipulating the link speed. This technique does not require firmware changes on involved network devices.
9) BRIGHTNESS [25] explores the use of a slight variation in the brightness of a display as a way of encoding data for exfiltration purposes. An adjustment of 3% in the red color component was used to encode a binary sequence on a 19" screen at a rate of 5 bit/s, making it possible to capture exfiltrated data at 6 m from the screen. The evaluation process proved that 5 to 10 bps data rates can be obtained at distances of 1 to 9 m, depending on the capture device that is used.
10) CTRL-ALT-LED [26] uses the keyboard LEDs to exfiltrate data from air-gapped computers. Three keyboard LEDs (CAPS, SCROLL, and NUM LOCK) are used to encode 3 bits of data (light represents 1 or 0 when it is either on or off), with the capture component using a video camera hidden on a smart watch of an employee or a compromised security camera. In certain cases, such as in Linux, the keyboard LEDs are accessible from user space, so no special permission is required. Transfer rates of 15 to 45 bps can be achieved, depending on the encoding technique, with tests confirming a transfer rate of 30 bps at a distance of 9.5 m.
11) aIR-Jumper [27] demonstrates how near infrared LEDs used to provide light vision illumination for surveillance cameras can be used as a covert channel, both for exfiltration or infiltration purposes, as the cameras can also be used to read data modulated in infrared pulses.

The use of wireless network covert channels is addressed by [28] and [29], which provide two takes on the subject: the first resorted to Bluetooth controlled light bulbs to implement both light and stenography-encoded payloads to create covert channels, while the second resorts to microjamming and backscatter modulation to exfiltrate data over ZigBee or WiFi channels.

Closer to our approach, which is focused on using the building automation infrastructure as a covert channel, there are considerably less examples. Tienteu et al. [30] demonstrated an exfiltration scenario using 802.15.4/ZigBee wireless communications, which is a consumer IoT solution and not a large scale, structured automation protocol, such as KNX or BACNet. A specific BACNet-based exfiltration scenario is demonstrated in [31].

While this section is by no means exhaustive, the aforementioned examples provide an encompassing perspective on the topic of data exfiltration from air-gapped devices and/or infrastructures, as well as the nature of the side channels already identified and successfully exploited in the past, as summarised in Table I, organised by method, medium, distance, and throughput. To the best of our knowledge, our article is the first reported example of using the KNX fieldbus [32] as a covert channel.

TABLE I
SURVEYED EXFILTRATION TECHNIQUES

Method	Medium	Source	Distance [m]	Speed [bps]	Refs.
Acoustic	Sonic	Speaker	∼30	20	[10]
		Computer fan	8	0.3	[15]
		HDD or Optical drive actuators	8	0.3	[16]
	Ultrasonic	Speaker	∼20	3	[9] [10] [14]
Electromagnetic		Video card/ cabling	∼8-30	480	[17]
		CPU/RAM bus	5+	1-2	[11]
		CPU	0.2/0.4/1	10/1/1 (loss=0)	[20]
		USB bus	9+	640	[18]
Thermal		CPU/GPU	∼0.4	0.13	[21]
		HVAC	room	0.83	[22]
Optical	Visible Light	Keyboard, HDD, LEDs, Lamps	line of sight	150	[6] [12] [24] [26]
		LCD Screen	8	20	[25]
	Laser	Laser, Scanners, Drones	1200	20	[23]
	Infrared	Surveillance cameras	10-100's	20	[27]
IoT, BACS, Backscatter sideband	Zigbee, WiFi	Implanted device	15	40	[29]
	Bluetooth	Light bulbs	25	120	[28]
	BACNet	Compromised device	n/a	n/a	[31]

Next, we provide an overview of the KNX framework for BACSs, so that the reader can grasp the technical details of the proposed approach.

III. KNX-BASED BACS

BACSs constitute a broad family of technologies that are used to automate the operations of buildings in terms of lighting, ventilation, heating, air conditioning and access control, among others. As such, traditional BACS are often associated with extensively automated buildings, managed in an integrated manner and supported by supervision control and data acquisition (SCADA) systems and domain-specific

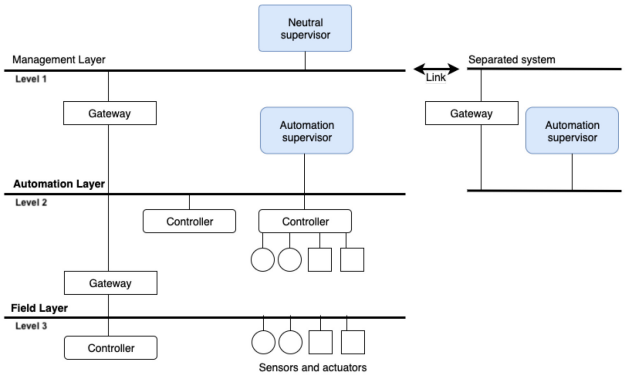

Fig. 1. Three-layer BACS Architecture (adapted from [36]).

standards and protocols, such as KNX [32], BACnet [33], and LonWorks [34].

ISO 16484 [35] specifies the various phases required in BACS projects and the hardware needed to perform the tasks within a BACS, as well as the requirements for overall functionality and communication. According to those specifications, the building automation framework is organised in three distinct layers: *Management*, *Automation*, and *Field* (Fig. 1). Within the scope of this article, the *field layer* is the most important, since it interconnects the sensing and control devices spread across the building and is often overlooked from a security management point-of-view.

The exfiltration techniques hereby proposed were designed to take advantage of this communication bus, with the KNX protocol being specifically adopted for the proof-of-concept implementation.

The KNX standard emerged in the early 1990s, driven by the European installation bus association (EIBA [32]) as a way to enable the connection, configuration, and communication between multiple building automation devices (e.g., sensors, actuators, buttons, and other user interfaces), using a standard communications protocol. It is widely used for home and building automation to control lighting, shutters, security systems, energy management, heating, ventilation, air-conditioning systems, signaling, and monitoring systems, remote control and audio/video control, among others. All these functions are managed via the KNX protocol set.

Unlike traditional electric installations, in KNX there are no direct hardwired connections between control and actuating devices. For example, a light switch is not directly connected with the controlled lights. Instead, all devices are connected via a shared bus that runs on 29 V. This means, for instance, that a light switch located inside a protected room will share its communications bus with lightning switches and actuators located in other rooms of the building.

Fig. 2 shows a simplified KNX system topology, that includes the following components: 1) power supplies that feed the bus and KNX devices; 2) sensors (switches, thermostats, air velocity meters, etc.) that generate commands as *telegrams or messages*; and 3) actuators (relays for lights, blinds, etc.) that receive the *telegrams* and perform predefined actions; and the bus that connects all devices.

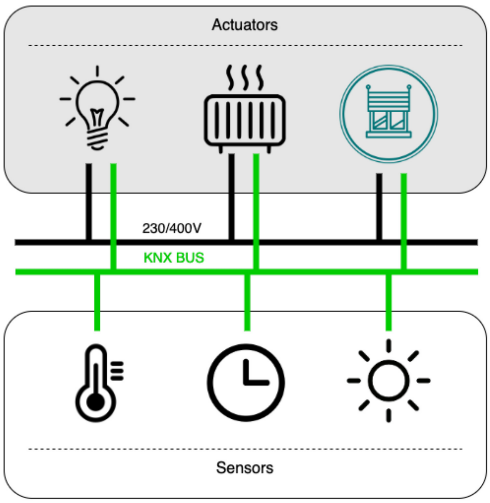

Fig. 2. Basic KNX elements.

octet 0	1	2	3	4	5	6	7	8	..	N - 1	N ≤ 22
Control Field	Source Address		Destination Address		Address Type; NPCI; length	TPCI	APCI	data/ APCI		data	FrameCheck

Fig. 3. KNX standard frame structure.

The most common transmission medium in KNX is twisted pair (TP). Other media are supported, such as powerline networking, radio frequency, infrared, and Ethernet/IP, but seldom found in production systems.

KNX messages are serially encoded in frames or telegrams, which are sent over the bus. Fig. 3 shows the structure of the logical protocol data unit (LPDU) of the KNX standard frame [37]. These LPDUs are the messages exchanged "above Layer 2", and include the following fields.

1) Control Field—determines the frame priority and distinguishes between Standard and Extended frames (where $N < 255$).
2) Source Address—the individual address (IA) of the source device.
3) Destination Address—either an IA for point to point communication or a group address (GA), in the case of multicast messages.
4) Address type—specifies the type of the destination address (IA or GA).
5) Hop Count—an integer decremented by routers, to avoid looping messages.
6) Length—the frame length.
7) Transport layer protocol control information (TPCI)—controls the transport layer communication relationships, for instance to build up and maintain a point-to-point connection.
8) Application layer protocol control information (APCI)—can tap into the full toolkit of application layer services (Read, Write, Response, ...,) which are available for the relevant addressing scheme and communication relationship.
9) Data—the message payload. Depending on the addressing scheme and APCI, the standard frame can carry up to 14 octets of data. This payload may be increased with the usage of extended frames.
10) Checksum.

At application level, the KNX protocol defines multiple functions, such as *Group Value Write* and *Group Value Read*, which are, respectively, used to send data from a given device to a group, and to receive data destined for a certain group. For instance, these functions are often used for a push button to send a message to turn a light on or off. Addresses are made up of two bytes, even though by convention they are often referred, in text form, as XX.YY.ZZ or XX/YY/ZZ (for IA and GA, respectively).

IV. Data Exfiltration Using KNX

Most of the work carried out in the field of data exfiltration is focused on the identification of potential side channels and analysing to which extend they may be successfully exploited. This work is no exception, as it started as an exercise to identify which systems and active devices could be present on air-gapped spaces, isolated from any type of communication and/or emissions (magnetic, optical, radio frequency, electromagnetic). Quite often, and due to specific confidentiality and security needs or even compliance with specific certifications (which are often mandatory for subcontractors), those isolated spaces are created within the premises of organisational building sites, which are often equipped with BACS. The basic premise for this article is to use the BACS infrastructure as a covert channel to exfiltrate data outside isolated spaces and, in some cases, of the building itself.

When compared to most of the exfiltration techniques presented in Section II, using the BACS infrastructure provides several advantages: this infrastructure is usually spread all over the building, making it easier to collect exfiltrated data (when compared with close proximity or line-of-sight requirements of other techniques); it may support bidirectional communications; it may support comparatively good throughput rates; and it can be combined sequentially with some of those techniques for multichannel/multihop exfiltration.

Section III presented the KNX reference architecture that is generally used in BACS/KNX deployments. In these deployments, sensors, switches, and actuators are interconnected by a fieldbus that consists of a TP cable. As the BACS permeates the entire building, typically there are several devices in each room, such as switches, fire detectors, presence detectors, or BACS-integrated climate control equipment. At each device installation point there will be a bus coupling unit (BCU) connected to the fieldbus that can be easily accessed, allowing the attacker to access the building's BACS network. The device used by the attacker to access the BACS network can be placed temporarily (e.g., during the time the attacker is alone in a room) or, more likely, can stealthily replace an existing device (e.g., replacing a regular light switch with a compromised light switch, nondiscernible to the human eye).

Following this rationale, two possible approaches to exploit the BACS field bus for data exfiltration are presented next:

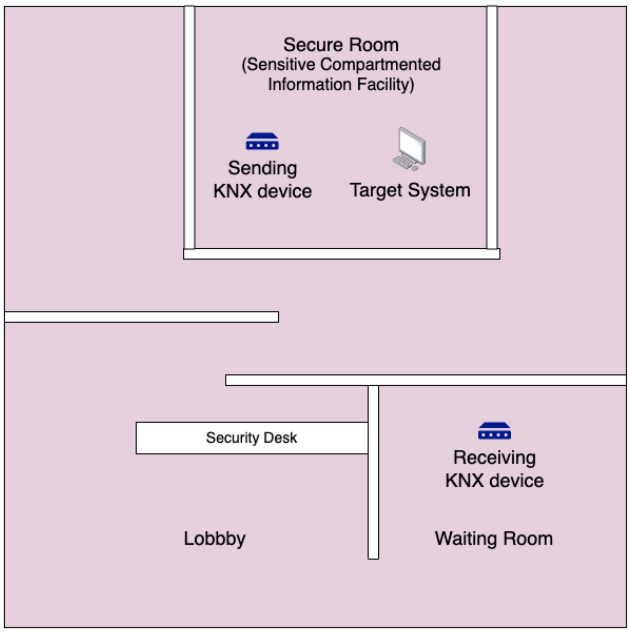

Fig. 4. Office plan.

sending data to another point of the BACS network, likely located in a less secure room where the attacker can collect the data; and activation of one or more luminaries (possibly visible from some point outside the building), for transmission of sensible data directly to the outside.

A. Exfiltrating Data Through the Fieldbus Network

The main idea of this approach is to send data through the BACS network, using one of the existing automation device connections as an entry point within the air-gapped room, and a second device located in another space (typically with easier access) as an exit point, therefore, allowing to receive the exfiltrated data (see Fig. 4).

How the data originally reaches the sending KNX device is outside the scope of this article. Nevertheless, several options can be considered.

1) The attacker is in the protected room, and uses for instance a USB port in the KNX device to directly connect it to the computer or storage device where the sensitive data is located.
2) The sending KNX device has for instance concealed microphones, which record private conversations in the protected room and send them, encoded, to the KNX receiving device.
3) The computer where the sensitive data is located has been previously compromised, and one of the short-range methods discussed in Section II (for instance [14], [16], and [17]) is used to move data from the (previously compromised) target system to the sending KNX device. This sequential use of two different side channels combines the benefits of short range channels (for extracting data from compromised systems) and longer range channels such as KNX, to make the location of the final receiving device much less restricted.

To send significant amounts of data across the network, the attacker must slice it into sets of bytes small enough to be encapsulated within valid KNX messages, to be later reconstructed at the reception point. These messages must also flow over the network in a stealthy way without interfering with normal building operation and without being detected. This implies understanding how to embed them in normal operation flows. For instance, the peer-to-peer communication messages used mostly in installation and commissioning actions should not be used, since they could trigger an alert as they could be associated to scouting operations.

The standard structure of KNX messages was already discussed in the previous section (cf. Fig. 3). The majority of messages used in production environments are sent from any device to a GA, using broadcast over the serial line. All devices, when commissioned, are programmed to listen for group messages related to their tasks. This architecture allows one-to-many communications, and assumes that when a given device hears a message that is not in its object database it will simply ignore it. Thus, if the messages used to exfiltrate data are addressed to any GA not used in the production environment, they will be ignored by all existing devices except the one used by the attacker as receiver.

Sending a message of this type requires the APCI field to be set for a *Group Value Write* message, that is, the binary value **0010**. The sliced data will then be encapsulated in octets 8 to 21 (a maximum of 14 bytes per frame)—with an empty payload being considered the end of file or stream transmission, for sake of simplicity, in the case of our proof-of-concept implementation.

The proof-of-concept implementation is backwards compatible with any KNX device, as it uses the *Standard Frame* data format. The use of *Extended Frames* could allow to send a larger payload per frame (up to a maximum of 254 bytes), increasing throughput. The KNX design would still allow these frames to be ignored by previous equipment, as they would soon be considered invalid in the control field analysis.

B. Using the Lighting System to Exfiltrate Data

The approach described in the previous section works well when it is feasible to install two compromised KNX devices (sending/receiving), and when the attacker has some sort of access to the receiving device, within the building, to eventually collect exfiltrated data.

However, in some scenarios, later access to the building (to install or access a receiving KNX-device) is not possible.

For those scenarios, an alternative solution is to use already existing and nonhacked building equipment, such as luminaries, as a replacement to the receiving device. For instance, the sending KNX device encodes sensitive data as commands to luminaries in other rooms (possibly visible from outside the building) using binary encoding, such as basic Morse code or more sophisticated schema. The induced blinking of these luminaries can be captured by cameras placed outside the building, allowing to reconstruct the transmitted data by decoding the blinking patterns.

Turning a luminary on and off via the KNX protocol consists of sending a *Group Value Write* message, with a value of **1** or **0**, respectively, addressed to the GA that is associated with an actuator object for that luminary. Hence, sending a batch of messages in a controlled manner allows sending data according to certain encoding rules. Moreover, the use of multiple luminaries in a synchronised way may allow for parallel transmission, increasing the throughput and/or enhancing stealthiness.

The KNX messages that are sent have a size of 8 bytes: one byte corresponds to the desired state (0 or 1), while the remaining bytes carry control information. It should be noted that the APCI will also be the binary value **0010**, corresponding to the *Group Value Write* function. Such messages are similar to those used to control luminaries in regular building operation, thus, hampering detection by stateless mechanisms (since the information is not encoded in the message payload, but rather in the on/off cadence, or in the conjunction between actions on different luminaries). Moreover, the use of an existing source address in the system will also allow for increased message obfuscation and overall stealthiness.

While the previous approach requires accessing the KNX fieldbus in two different locations (to send and to receive sensitive data), this approach only requires accessing the fieldbus to send the sensitive data, since no changes are necessary at the luminaries. Moreover, this makes it possible for the receiver to be outside the building—which in some cases compensates for the lower data rates achievable with this alternative.

To improve stealthiness, several strategies can be adopted. The most basic is to resort to light fixtures placed in spaces where the blinking would not call the attention of building occupants, or to adjust the blinking frequency to make it imperceptible to the human eye (as explored by [12], in a different scope).

A more stealthy approach may be implemented if a dimmer is present. Dimming luminaries in small steps (instead of using on/off commands) may be imperceptible to humans (as demonstrated in [25], for LCD screens). Curiously, several KNX dimmers also support a specific load mode that allows them to control for instance solenoid valves or single phase motors (often used to drive ceiling fans or active ventilators). This could eventually pave the way for the implementation of exfiltration methods akin to those proposed in [15].

Specifcally for KNX, one must take into account that dimming control disciplines are pre-established via the ETS tool, which allows to configure parameters related to dimming control (for instance, whether there is a looped single button or two buttons for control), the delays between switching and dimming or the dimming percentages per step. Even if we discard more complicated strategies, such as compromising engineering stations or taking possession of BCU keys (when used to protect devices), which would provide an attacker with the means to fine tune dimming control via provisioning mechanisms, attacks may be implemented via inline control mechanisms. In fact, if enough dimming levels are supported (via small steps), an attacker may generate write messages (akin to the ones produced by a two-button up/down controller) to slightly increase and decrease brightness to modulate data transfers, in a more stealthy way. Finally, this concept could eventually be improved if we consider that many KNX dimmable LED drivers also support RGB led strips, meaning that color-based encoding could be possible (as shown in [28]).

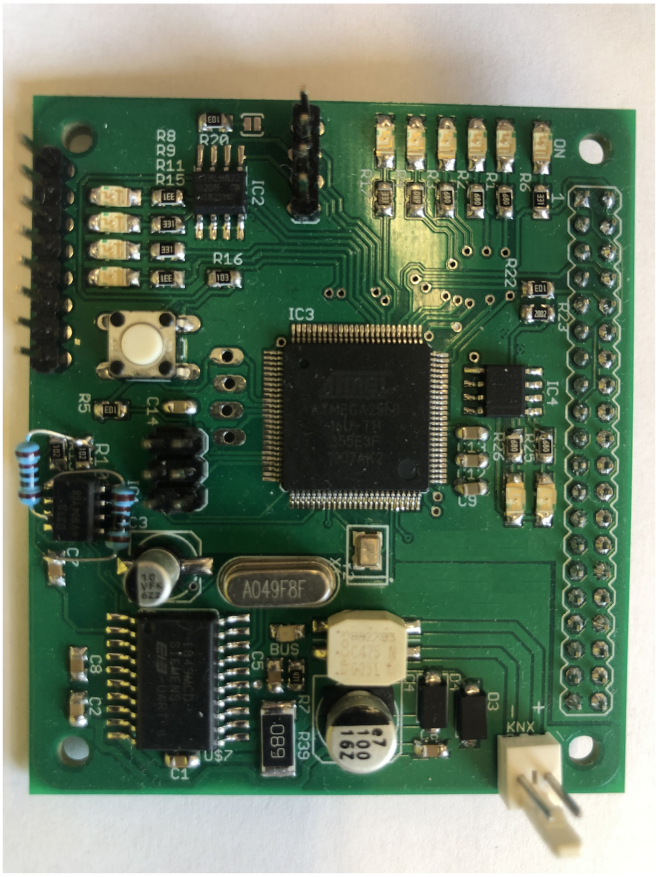

Fig. 5. KNX TP shield—BCU.

V. Proof-of-Concept Implementation

In this section, we describe the proof-of-concept implementation of the two proposed exfiltration techniques: 1) inline (via KNX bus) and 2) via light dimming.

A. Inline KNX Bus Exfiltration

A Raspberry Pi 4 was adopted as base SBC (small board computer) platform for the KNX devices, due to its availability and low cost, even though real-world attacks would likely resort to devices with smaller footprints and disguised as typical KNX components.

A KNX BCU was developed in the form of a shield (see Fig. 5) connected to the SBC GPIO interface. This shield contains a TP-UART (optically isolated from the SBC host using an ILD213T optocoupler) that provides fieldbus connectivity, as well as an ATmega 2560 micro controller that establishes the interface between one of the SBC serial ports and the TP-UART serial line (see Fig. 6).

We used a Debian Linux distribution (Raspbian) and the GO language to develop the device. The GoPacket open source library was used [38] for decoding and encoding messages.

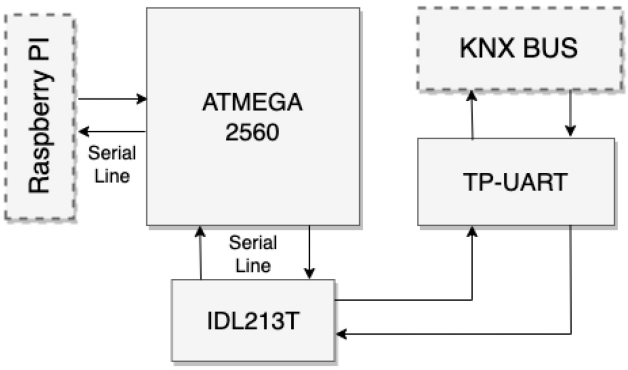

Fig. 6. Shield diagram.

Since this library originally had no support for KNX, we extended it to support manipulation of KNX protocol flows.

Apart from the KNX-TP shield (which was integrated into a single printed circuit board), the whole prototype was built using commercial of-the-shelf hardware components, with no special optimisation. The SBC could be stripped from unnecessary components, eventually being replaced by a Raspberry Pi *Compute module*. Also, an increased component integration on the shied could be achieved, along with the removal of components used for debugging, like LED's, push buttons and LCD connections. This means that the physical device form factor could take a fraction of the prototype footprint and could certainly be resized to the thickness of a credit card.

In the scope of this work, and regarding the proposed exfiltration techniques, two command line applications were developed (*KNXsend* and *KNXreceive*). Together, when used in devices connected to different points of the KNX bus, they allow sending and receiving a file, thereby creating a covert channel for data exfiltration.

As already mentioned in Section IV, it should be noted that this article is mostly focused on demonstrating how the KNX network could be exploited to exfiltrate data out of an isolated/air-gapped space, not considering the physical intrusion/access process or how the target data was obtained in the first place, as such matters are outside the scope of the present work.

The first thing to do after gaining access to the restricted/secure physical space is to locate a KNX component, replacing it by the prototype device, which will be connected to the fieldbus, to enable data transmission. A counterpart device must also be discreetly deployed in a place with less strict access, in order to receive the transmitted data.

First, the *KNXreceive* application is configured to receive messages from the sender with IA address 1.1.4, and destined to the GA address 5/5/5. The device connected to the bus will (stealthily) listen to any messages traveling through the bus using the above identified source/destination pairs. Second, the data file to be transmitted is provided to the *KNXsend* application, which will slice it into byte blocks, for encapsulation in KNX messages using IA 1.1.4 as source and GA 5/5/5 as destination.

The developed code uses an internal buffer with a maximum of 14 bytes, in order to guarantee that the data sent in each message can use standard KNX frames. Depending on the building, the payload size could be increased with the use of extended KNX frames, therefore, enabling to use up to 254 byte payloads.

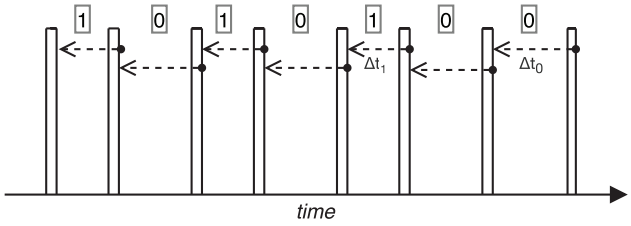

Fig. 7. D-PPM modulation.

B. Dimmer-Based Exfiltration

The implementation of the exfiltration method described in Section IV-B only requires one device, connected in a similar way at the location from where the data is accessed and exfiltrated from. Prior scouting work would be carried out to identify the KNX GAs associated with the intended (already existing) luminaries. Finally, the sending application would generate dimming commands (or on/off messages, in alternative) at controlled time intervals to activate the luminaries, whose dimming (or blinking) patterns would be captured by video cameras.

For this specific case, one possible approach may be based on a low-rate transmission strategy (compatible with a KNX control flow rate), for instance, sending bits by means of Differential Pulse Position Modulation (D-PPM). D-PPM encodes bits as sequences of pulses with a variable gap (see Fig. 7). This way, the pulse position is encoded in such a way that the receiver must only measure the difference in the arrival time of successive pulses to distinguish between zeros and ones.

In order to undertake a feasibility evaluation study, a proof-of-concept device was built using an Arduino Uno microcontroller, which communicates with the SBC via I2C. This Arduino is coupled to an LED driver circuit using three TIP120 transistors, connected to a 12V WS2812 LED strip. The implementation for the modulation side follows Algorithm 1. For brightness control, we used the pulse width modulation (PWM) capabilitiy of the ATMega microcontroller, which was deemed suitable for this application since most LED strip dimmers use PWM, with variable duty cycles to control LED brightness levels.

In the specific case of this proof-of-concept implementation, two brightness levels were configured: $\text{level}_{\text{low}}$ and $\text{level}_{\text{high}}$, corresponding to 90% and 100% duty cycle modes. The difference between them is practically indistinguishable to the human eye, but can be easily detected by a smartphone camera.

The demodulation component (which could be located outside the building, as long as there is line of sight to the LED strip) may run in a smartphone or a dedicated device. In the specific case of the proof-of-concept implementation, Python was used, along with the *opencv*, *pandas*, *scipy*, and *numpy* libraries.

Algorithm 1 D-PPM Modulation

Pre-established: *bitdelay*, the impulse duration; *zerodelay*, the duration for a binary "0"; *onedelay*, the duration for a binary "1". Brightness levels adjusted for two-step dimming (*high* and *low*).

```
procedure BITCODE(boolean bit)
    brightness(level_high)
    delay(bitDelay)
    brightness(level_low)
    if bit == 0 then
        delay(zerodelay)
    else
        delay(onedelay)
    end if
end procedure
```

Algorithm 2 showcases the demodulation process, which is performed from a captured video file. First, the user needs to define a region of interest (ROI) in the video stream, in order to speed the decoding process and avoid unwanted noise sources—for this reason, the video capture must be performed from a stable device in order to make sure the ROI is not shifted over time. Subsequently, all the video frames are preprocessed via cropping (to ROI boundaries), converted to grayscale, and subject to a average blur filter with a 6×6 kernel before being averaged. The brightness averages for each frame are stored on the *avgvalues* list.

Once all frames are processed, the first derivative of the resulting time series is calculated. The gradient eases the task of locating the rising edges of the PPM pulses (detection thresholds are calculated accordingly with the algorithm, which was experimentally adjusted to 1/3 of the time series value range). From this point on, it is just a question of locating the timestamps for each edge and computing the time deltas between consecutive edges. The results obtained with our proof-of-concept implementation of the D-PPM dimming schema will be discussed in Section VI.

VI. Validation

The validation of the proposed techniques was carried out in two different scenarios. First, a laboratory testbed allowed the definition of a performance baseline for inline bus exfiltration, also providing the integration context for the dimming scenarios. Second, a single-family dwelling allowed validating inline bus exfiltration in a production environment, to assess potential mutual interference between the exfiltration process and the regular operation of the building.

A. Laboratory Testbed

Fig. 8 shows the laboratory testbed that was created to test and assess the proof-of-concept implementation.

The following KNX devices were used in the testbed (see Fig. 9).

1) A USB/KNX TP gateway that allows connection to the KNX TP fieldbus (green cable).

Algorithm 2 D-PPM Demodulation

INPUT: Video capture file

```
avgvalues ← ()
bitstream ← ()
blur_ksize ← (7, 7)
sample ← LOADFRAME()
ROI ← SELECTROI(sample)

for each f ∈ frames do
    cropped_f ← CROP(f, ROI)
    gray_f ← GRAYSCALE(cropped_f)
    blurred_f ← AVERAGEBLUR(gray_f, blur_ksize)
    avg ← AVERAGE(blurred_f)
    avgvalues.APPEND(avg)
end for

grad ← gradient(avgvalues)
max_g ← max(abs(grad))
min_g ← min(abs(grad))
threshold ← −(min_g + (max_g − min_g)/3)
timestamps ← grad.WHERE(gradient > threshold)

for each t ∈ t_frames do
    deltas ← timestamps.DIFF(t, t + 1)
end for

for each d ∈ deltas do
    if d > 400ms then
        bitstream.APPEND(0)
    else
        bitsteam.APPEND(1)
    end if
end for
```

OUTPUT: *bitstream*, Demodulated bit stream

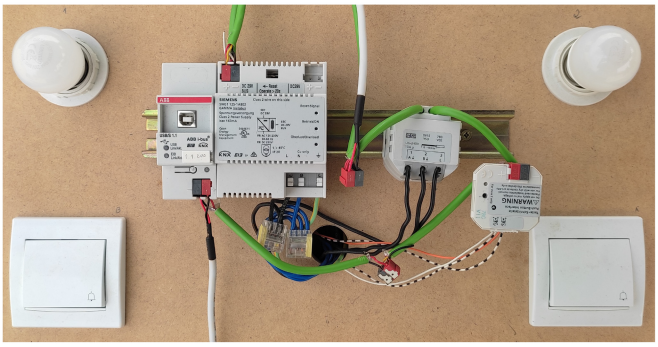

Fig. 8. KNX testbed.

2) A power supply that guarantees the fieldbus power supply to the intended 29 V.
3) An actuator that allows the connection of two normal switches to the system, allowing their operation.
4) A relay module that receives information from the KNX bus and controls the lamp operation.

Fig. 10 depicts the wire schematic of the laboratory testbed, including power lines and field bus lines. There is total

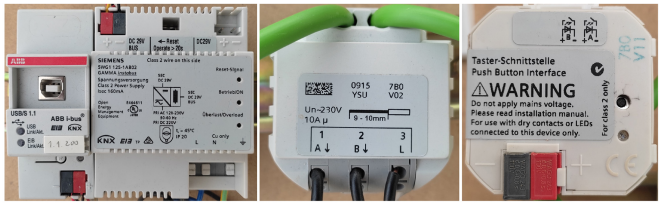

Fig. 9. USB gateway/power supply/relay/actuator.

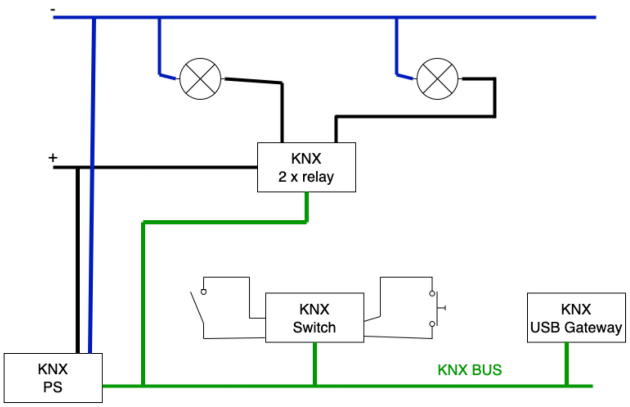

Fig. 10. KNX testbed—wire schematic.

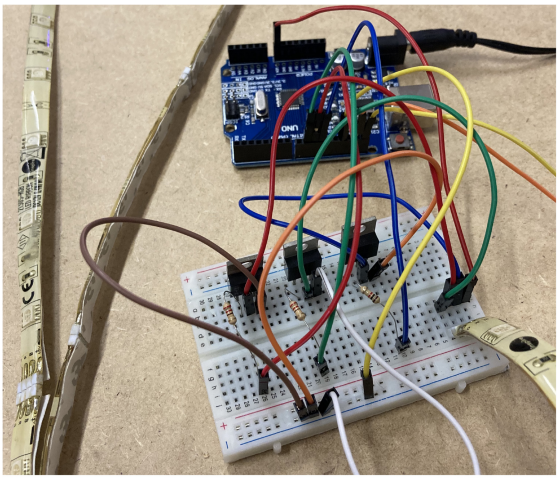

Fig. 11. LED strip driver.

independence between the actuators/sensors and the relays that control the equipment (e.g., lamp) operation. The fieldbus (the KNX TP) transports messages between the various devices, therefore, enabling building operation. When a switch is activated, a GA message is sent to the network and the relay, which has as been configured to receive this message, will activate in order to turn the light on/off in the required channel.

The laboratory testbed also hosts the custom LED strip driver, depicted in Fig. 11. This driver will be used for the dimming exfiltration tests.

This simple installation provides the means to replicate most of the operations existing in a BACS. It can be configured, commissioned and debugged through the KNX engineering tool software (ETS), using the USB gateway. It also provides connection for one or multiple KNX devices like those built to host the *KNXsend* and *KNXreceive* applications.

B. Production Environment

Additionally to the laboratory environment, tests were carried out in a single-family house with three floors. This house is equipped with a KNX home automation system that controls the lighting, blinds, the heating system, and the alarm system. Some local environments are also controlled through light and motion detectors. The building has devices for monitoring the environmental conditions outside, as well as movement and lighting control in the house backyard and its annexes. This three-floor house includes an entrance hall, a library, two living rooms, four bedroom suites, a kitchen, a pantry, a laundry room, two offices, and a few technical and storage areas. The existing home automation system can be summarized in the following points.

1) There are power and control cabinets in each floor.
2) A KNX twisted-pair fieldbus allows communication between all the devices.
3) Regarding the lighting system, all the luminary power cables are routed to the control cabinet where the actuators are located, with sensors and switches located in each of the rooms.
4) Central heating is guaranteed to each room by temperature sensors and individually controlled solenoid valves. The individual control of each radiator is enabled by actuators located in the control cabinet, thus, guaranteeing the possibility of different environments in each room of the house.
5) The shutters are electrically operated from the control cabinet and, locally, there are switches that allow selection of the desired functions.
6) Motion and light sensors are used with the dual function of controlling the environment and the alarm system.
7) Centralized control is also possible, enabled by an existing touch panel. This device offers individual or grouped acting of the different equipment's. The use of timers and actions resulting from previously established conditions are also allowed.

This house can be compared to an office building in terms of the extension of its automation network and the diversity of existing equipment, so it was used to validate the described attack scenario. In this scope, a storage room was used as the room from where sensitive data had to be exfiltrated. This room does not have any fenestration, however, it has lighting and the respective KNX switch. A room located next to the main entrance was used as the (supposedly more accessible) place where the attacker's accomplice would be receiving the exfiltrated data.

C. Inline Exfiltration Tests

The inline exfiltration tests performed in both scenarios (testbed, production environment) consisted of sending files with content randomly generated from */dev/random*, with the following sizes: 1 K, 10 K, 100 K and 1 M bytes. The tests were repeated ten times and the total transmission and reception times, as well as the bytes transmitted and received, were registered. It was confirmed that the received content matched the sent content.

TABLE II
DATA EXFILTRATION PERFORMANCE—LAB TESTBED

File Size	Direction	Mean latency [s]	Std [s]	Throughput [bps]	Packet loss [%]
1K	send	11,489	0,0003	713	0
	receive	11,581	0,0003	707	
10K	send	115,046	0,0031	712	0
	receive	115,149	0,0033	711	
100K	send	1151,065	0,0261	712	0
	receive	1151,179	0,0255	712	
1M	send	11787,245	0,1632	712	0
	receive	11787,511	0,1688	712	

TABLE III
DATA EXFILTRATION PERFORMANCE—PRODUCTION ENVIRONMENT

File Size	Direction	Mean latency [s]	Std [s]	Throughput [bps]	Packet loss [%]
1K	send	11,492	0,0057	713	0
	receive	11,563	0,0055	708	
10K	send	115,086	0,0312	712	0,014
	receive	115,170	0,0316	711	
100K	send	1151,361	0,0832	712	0,017
	receive	1151,456	0,0847	711	
1M	send	11791,693	1,9070	711	0,019
	receive	11791,947	1,9082	711	

Table II shows the measurements obtained in the laboratory testbed. A peak rate of 712 bps was obtained in this controlled environment, with no packet losses. It should be noted that the only messages transmitted on the bus—during the execution of these tests—were those related to the ongoing data exfiltration. The standard deviation for the measurements is very small, which is explained by the stability of the field bus. Variations were proportional to the size of transmitted files and, consequently, to the number of transmitted packets.

The measurements obtained in the production environment (house scenario) are provided in Table III, and denote a higher standard deviation, that results from the impact of the regular KNX traffic that is present in the field bus. Nevertheless, measured throughput is identical, suggesting that for small office buildings the impact of fieldbus traffic is not significant. The measured packet loss is very low but not zero, which suggests real world attacks definitively need to support recovery from packet losses (as was the case of our proof-of-concept). Building occupants did not report any strange behavior from the building's automation system, which is a good indicator of the stealthiness of the proposed technique—moreover if we consider that some of the tests lasted longer than 3 h.

D. Visible Light Dimming Exfiltration Scenarios

For evaluation purposes, the dimming driver controller was deployed in two different setups (see Fig. 12): 1) a close range setup, within the laboratory; and 2) an outdoors scenario. Straight line distances were approximately 4 and 15 m, respectively. In both cases, the ASCII-encoded string "ABC" was continuously sent, in a loop. Video capture was performed

Fig. 12. Dimming test setups (red rectangles locate the light sources).

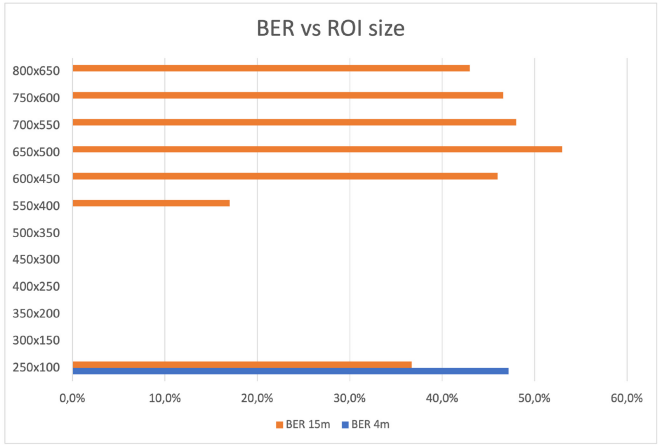

Fig. 13. Bit error rate (BER) versus ROI size.

using a second generation Apple iPhone SE, in 1080p HD mode, at 30fps.

Figs. 13–15 present the results obtained with the decoding algorithm presented in Section V-B. The tradeoff between ROI dimension and demodulation accuracy was studied for several boundary sizes, starting with a 250 × 100 rectangle centred in the light source, increasing the width and height in steps of 50 pixels. Results are shown in Fig. 13, demonstrating that in our case the optimum ROI (no observed errors) was between 300 × 150 and 500 × 350.

As expected, the correct choice of the ROI area seems to be particularly relevant for the outdoors scenario. Most likely this is due to the fact that a balanced ROI boundary size may filter some noise while, at the same time, allowing for optical phenomena (such as light scattering, which may vary accordingly with room and window characteristics) to enhance the detection rate. Camera setup is also crucial for a good capture: optical artefacts introduced by auto-focus and automatic sensitivity adjustment greatly affect the demodulation process, as they induce brightness distortion.

Figs. 14 and 15 provide a visualization of the time series gradients which were used for D-PPM demodulation purposes. Red lines depict the interpulse widths corresponding to the time deltas between consecutive bits. While threshold

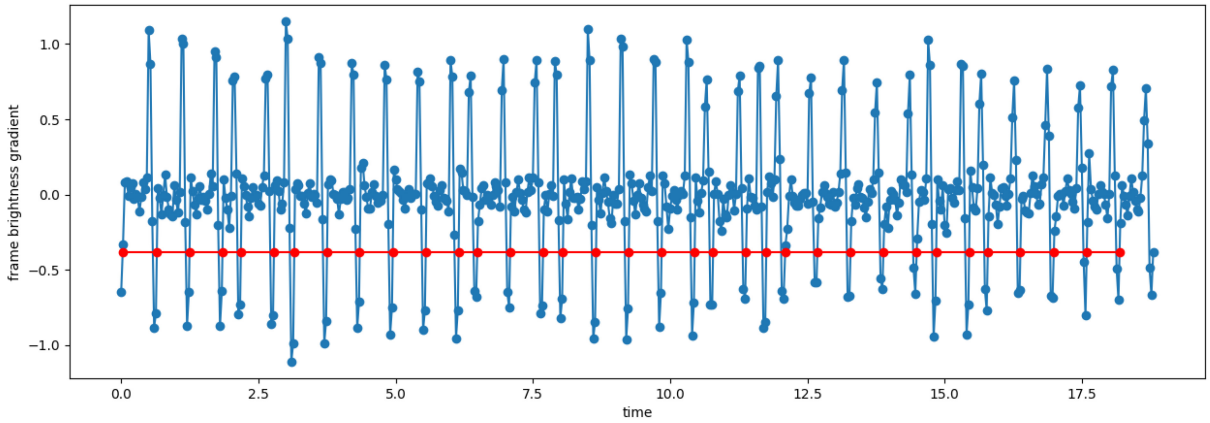

Fig. 14. Results for 4 m indoor test, 350 × 200 ROI area.

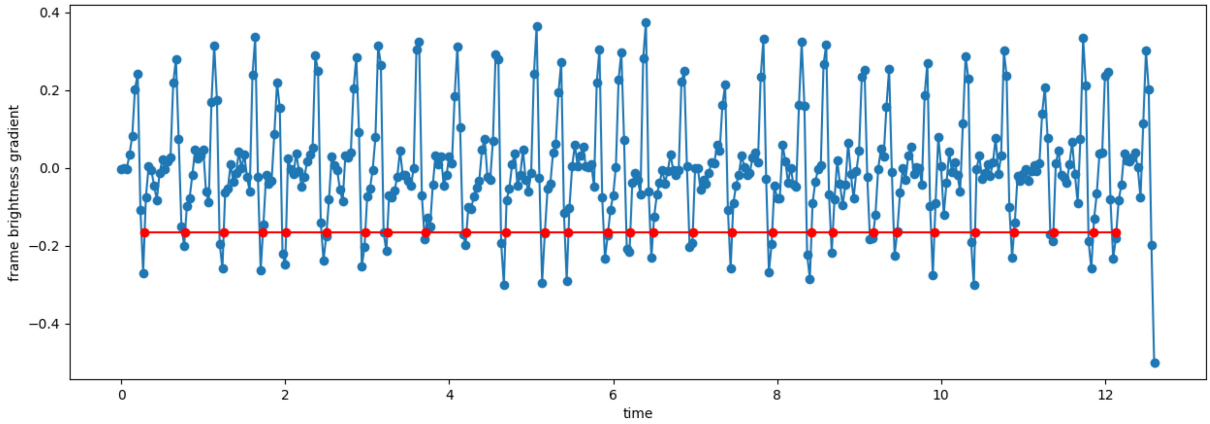

Fig. 15. Results for 15 m outdoor test, 350 × 200 ROI area.

detection is undertaken using a simple heuristic (as shown in Algorithm 2), in many cases a manual adjustment may be required to improve sensitivity.

For this proof-of-concept implementation, transmission speeds around 2 bps were measured. Despite being much lower than the first exfiltration method, these rates are still within typical ranges observed for more exotic side channels. Moreover, there is a good margin for improvement by means of finely tuning delay, as well as the introduction of more evolved modulation techniques and better signal processing, for noise filtering and thresholding.

VII. Possible Countermeasures

In this section, we discuss possible countermeasures for the exfiltration techniques proposed in this article, which take advantage of the fact that the KNX fieldbus is typically overlooked when analysing IT security but still widespread across the building, with relatively easy physical access. This is further reinforced by the fact that this fieldbus is often air gapped and not connected to a local network or the Internet, which keeps it outside the radar of IT staff.

A. Native BACS Mechanisms

Besides simply excluding KNX devices from sensitive locations, one could also consider implementing mitigation measures based on native mechanisms, such as building management systems (BMSs), line couplers or gateways, with varying degrees of efectiveness against exfiltration. For instance, the presence of conventional BMS was found to make little or no difference for security purposes in [39], as no abnormal situations were flagged during attack deployment. This is hardly unsurprising if we consider that most implementations do not include the required detection capabilities that could be used to leverage the BMS profile and role for security purposes.

In addition to galvanic isolation, KNX line couplers may provide filtering for busload reduction (pretty much similarly to Ethernet bridges in the early days of shared medium implementations). This latter function is supported by filtering tables filled using the ETS tool. According to defined configurations, it can be used to completely block telegrams (providing isolation), to provide full routing between topology sections or, by standard behavior, to filter telegrams according to the GA information. Due to its stateless nature, line couplers in default or full routing configurations offer little protection against telegrams specifically crafted to reach specific topology sections, meaning that the documented exfiltration techniques are still feasible, eventually requiring specific adaptations. Moreover, while line couplers could be used to implement whitelisting techniques, there is the risk of exceeding the device table memory capacity (especially, true for older equipment

TABLE IV
STRUCTURE AND FIELD VALUES FOR IDS RULES [39]

action	protocol	source address (SA)	source port (SPort)	operator	destination address (DA)	destination port (DPort)

`pass`
`alert`
`log`
	any
	knxtp
	KNXTP
		any
		IA
			any	$->$
				<>
					any
					IA
					GA
						any

or complex scenarios), with a variable degree of effectiveness (eventually none at all, for attacks within the same KNX line domain).

KNX gateways would be mostly ineffective against the documented attacks, as these are often used for the interconnection of different physical media and/or protocols (e.g., KNX TP-RS232 gateways and KNX-DALI gateways). However, the proposed exfiltration techniques use standard protocol addressing, framing and semantics for transport guarantee over a KNX line bus, using two purpose-built devices as endpoints for inline exfiltration and one for the dimmer attack. For these cases, which do not involve crossing different BACS domains, KNX gateways would be largely ineffective.

B. Introduction of Domain-Specific NIDS

Besides these BACS-native mechanisms, all other countermeasures require monitoring the field bus traffic, in order to detect anomalous behaviors. However, monitoring the KNX fieldbus traffic is not a common practice, which further increases its potential for stealthy data exfiltration. Moreover, simple monitoring techniques may not be enough. For instance, the sole use of volumetric detection techniques (e.g., based on traffic volumes or stateless packet inspection) might be rendered ineffective, since attackers may hide data exfiltration in telegrams resembling normal building control operations, and may also adjust the exfiltration throughput to keep it below traffic volume thresholds. This problem is well known in other environments (such as local IP networks), and some lessons may be potentially translated to and reused in KNX scenarios. Nevertheless, it is always necessary to adapt them to the specific nature of the KNX fieldbus and the messages it conveys.

To the best of our knowledge, there are no available security monitoring tools for the KNX fieldbus, with the notable exception of the network and intrusion detection system (NIDS) concept we proposed in a recent paper [39]. This KNX NIDS was developed with several attack scenarios and capabilities in mind, as discussed next.

1) Signature-Based Detection: The KNX NIDS supports signature-based detection, leveraging a knowledge base with rule definitions. Having been developed for detecting abnormal activity, such as device or bus scans, is also effective for inline exfiltration attempts involving messages destined to a specific GA or addresses outside the range of those being used in the system. This BACS NIDS uses a rule format whose fields are listed in Table IV.

For any match, the `pass` action ignores the message, the `alert` action issues an *alert* and the `log` action exports that message to a PCAP file. The `protocol` field matches a specifc protocol (e.g., KNX), with the operators allowing to specify if the rule applies in one or both directions. This syntax allows expressing complex rulesets, enhancing detection by means of white or blacklisting techniques. For example, the following whitelisting ruleset template is effective against inline bus exfiltration.

```
alert knxtp !<legitimateIA> any -> any any
(msg:"Unauthorized communication peer"; rid:31; rev:1;)

alert knxtp any any -> !<legitimateIA> any
(msg:"Unauthorized communication peer"; rid:32; rev:1;)
```

Instantiating this template for the existing device set would allow to whitelist the complete installation, detecting inline exfiltration attacks using unknown IAs. More complex rules could also be created by adding application-level fields to search for specific operations. Moreover, it should be noted that normal system operation is based mostly on application-level messages such as *A_GroupValue_Write_PDU*. This way, rules such as the one provided next could be added to detect abnormal group communications.

```
alert knxtp any any -> any any (
msg:"Application Layer - Device Info";
apci:!"A_GroupValue_Write_PDU"; rid:34; rev:1; )
```

For such an approach, the NIDS detection accuracy can potentially reach a perfect score, depending on the granularity of the whitelist rules. However, this technique may me intensive in terms of manual configuration, requiring extensive knowledge about specific setups.

2) Machine Learning Detection Techniques: The KNX NIDS we proposed in [39] also supports machine learning techniques, in order to reduce the management overhead and to enhance detection capabilities for abnormal sequential activation patterns.

For instance, in cases where it is common for the activation of lights to be driven by motion sensors, the proposed solution makes it possible to detect invalid context situations—in other words, commands sent to luminaries in rooms were there is no movement detection (i.e., no occupants).

While these heuristics can be easily encoded using rulesets, this aproach quickly becomes unpractical for medium and large-sized buildings without resorting to artificial intelligence techniques such as decision trees. We have demonstrated that logistic regression and neural networks provided excellent results for detecting attacks very similar to the proposed exfiltration techniques, in terms of traffic patterns [39]. While for device scouting attacks the scores were perfect, invalid context attacks (which involve device activation sequences not observed in normal operation) are also very good, as shown in Table V.

Other techniques potentially effective against the proposed exfiltration techniques could rely on statistical traffic network analysis, time series or timed Markov chains, which could be leveraged to detect and identify abnormal traffic patterns which may correspond to ongoing malicious activities. Modeling approaches based on the extraction of protocol interaction patterns, for instance using Discrete Time Markov Chains

TABLE V
INVALID CONTEXT ATTACK—BACS NIDS RESULTS [39]

Model	AUC	CA	F1	Precision	Recall
Neural Network	1.000	1.000	1.000	1.000	1.000
Logistic Regression	0.997	0.998	0.998	0.998	0.998

or Finite State Automata [40], [3], may also prove to be interesting. Overall, the proposed BACS NIDS is able to support and assist the development of these capabilities by providing the necessary building blocks, as it is the case for feature extraction mechanisms.

VIII. CONCLUSION

This article demonstrated how BACS can be abused for the purpose of exfiltrating sensitive data, constituting a potential security risk for organisations, moreover if we consider that such data theft can be carried out in a stealthy way (as it was hereby demonstrated).

Evaluation results for the proposed proof-of-concept prototype have shown a measured transfer rate of 711 bps over the KNX field bus, with no packet loss for a controlled environment and negligible packet loss for a production BACS. This is more than enough for timely transmission of relevant information. Knowing that, for instance, a Bitcoins private key takes only 256 bits or that a SSL private certificate normally occupies 2048 bits, it would take only 3 or 24 s, respectively, for its exfiltration. These figures have room for improvement, as the validated prototype did not implement compression, error correction or other methods of potential optimisation, paving the way for future research and development directions. Moreover, this exfiltration technique can be bidirectional, allowing attackers to insert malicious code/-data in the target systems or to control them from a remote Command & Control Server.

A second method involved sending KNX commands to building luminaries as a means to access exfiltrated data from outside the building. While achieved data rates were considerably lower, as expected, they are still within the usable range for many practical use cases (e.g., exfiltration of passwords or certificates), and there is also considerable room for improvement.

Finally, it should be pointed out that this work provides a perspective on a side channel that not been yet explored (at least to the authors knowledge), contributing to call the attention of the industrial and research communities to the importance of researching and developing adequate protection measures, and also calling for the introduction of proper regulation and policies to protect both new and existing BACS deployments from potential abuse.

REFERENCES

[1] *Intelligence Community Directive 705—Physical and Technical Security Standards for Sensitive Compartmented Information Facilities*, Office Director Nat. Intell., Washington, DC, USA, 2010.

[2] *Technical Specifications for Construction and Management of Sensitive Compartmented Information Facilities—IC Tech Spec—For ICD/ICS 705—Version 1.5.1*, Nat. Counterintelligence Security Center, Washington, DC, USA, 2021.

[3] V. Graveto, T. Cruz, and P. Simões, "Security of building automation and control systems: Survey and future research directions," *Comput. Security*, vol. 112, Jan. 2022, Art. no. 102527. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404821003515

[4] A. Antonini, F. Maggi, and S. Zanero, "A practical attack against a KNX-based building automation system," in *Proc. 2nd Int. Symp. ICS SCADA Cyber Security Res.*, 2014, pp. 53–60. [Online]. Available: https://doi.org/10.14236/ewic/ics-csr2014.7

[5] M. Guri and Y. Elovici, "Bridgeware: The air-gap malware," *Commun. ACM*, vol. 61, no. 4, pp. 74–82, Mar. 2018. [Online]. Available: https://doi.org/10.1145/3177230

[6] J. Loughry and D. A. Umphress, "Information leakage from optical emanations," *ACM Trans. Inf. Syst. Security*, vol. 5, no. 3, pp. 262–289, 2002.

[7] R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*. Indianapolis, IN, USA: Wiley, 2020.

[8] R. J. Anderson and M. G. Kuhn, "Soft tempest—An opportunity for NATO," in *Proc. Protecting NATO Inf. Syst. 21st Century*, 1999, pp. 1–5.

[9] M. Hanspach and M. Goetz, "On covert acoustical mesh networks in air," *J. Commun.*, vol. 8, no. 11, pp. 758–767, 2013.

[10] B. Carrara and C. Adams, "On acoustic covert channels between air-gapped systems," in *Proc. Int. Symp. Found. Pract. Security*, 2014, pp. 3–16.

[11] N. Matyunin, J. Szefer, S. Biedermann, and S. Katzenbeisser, "Covert channels using mobile device's magnetic field sensors," in *Proc. Asia South Pac. Des. Autom. Conf. (ASP-DAC)*, Jan. 2016, pp. 525–532.

[12] A. Robles-Durazno, N. Moradpoor, J. Mcwhinnie, and G. Russell, "WaterLeakage: A stealthy malware for data exfiltration on industrial control systems using visual channels," in *Proc. IEEE Int. Conf. Control Autom. (ICCA)*, Jul. 2019, pp. 724–731.

[13] M. T. Naz and A. M. Zeki, "A review of various attack methods on air-gapped systems," in *Proc. Int. Conf. Innov. Intell. Inform. Comput. Technol. (3ICT)*, 2020, pp. 1–6.

[14] M. Guri, Y. Solewicz, and Y. Elovici, "MOSQUITO: Covert ultrasonic transmissions between two air-gapped computers using speaker-to-speaker communication," in *Proc. IEEE Conf. Dependable Secure Comput. (DSC)*, 2019, pp. 1–8.

[15] M. Guri, Y. Solewicz, A. Daidakulov, and Y. Elovici, "Fansmitter: Acoustic data exfiltration from (speakerless) air-gapped computers," 2016, *arXiv:1606.05915*.

[16] M. Guri, "CD-LEAK: Leaking secrets from audioless air-gapped computers using covert acoustic signals from CD/DVD drives," in *Proc. IEEE 44th Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, 2020, pp. 808–816.

[17] M. Guri, G. Kedma, A. Kachlon, and Y. Elovici, "Air hopper: Bridging the air-gap between isolated networks and mobile phones using radio frequencies," in *Proc. 9th IEEE Int. Conf. Malicious Unwanted Softw. (MALCON)*, 2014, pp. 58–67.

[18] M. Guri, M. Monitz, and Y. Elovici, "USBee: Air-gap covert-channel via electromagnetic emission from USB," in *Proc. 14th Annu. Conf. Privacy Security Trust (PST)*, 2016, pp. 264–268.

[19] M. Guri, "BeatCoin: Leaking private keys from air-gapped cryptocurrency wallets," in *Proc. IEEE Int. Congr. Cybermatics IEEE Conf. Internet Things Green Comput. Commun. Cyber Phys. Soc. Comput. Smart Data Blockchain Comput. Inf. Technol. iThings/GreenCom/CPSCom/SmartData/Blockchain/CIT 2018*, Jun. 2018, pp. 1308–1316.

[20] M. Guri, B. Zadov, and Y. Elovici, "ODINI: Escaping sensitive data from faraday-caged, air-gapped computers via magnetic fields," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 1190–1203, 2020.

[21] M. Guri, M. Monitz, Y. Mirski, and Y. Elovici, "BitWhisper: Covert signaling channel between air-gapped computers using thermal manipulations," in *Proc. Comput. Security Found. Workshop*, Sep. 2015, pp. 276–289.

[22] Y. Mirsky, M. Guri, and Y. Elovici, "HVACKer: Bridging the air-gap by attacking the air conditioning system," 2017, *arXiv:1703.10454*.

[23] B. Nassi, A. Shamir, and Y. Elovici, "Oops!··· I think I scanned a malware," 2017. *arXiv:1703.07751*.

[24] M. Guri, "Optical covert channel from air-gapped networks via remote orchestration of router/switch LEDs," in *Proc. Eur. Intell. Security Inform. Conf. (EISIC)*, 2018, pp. 54–60.

[25] M. Guri, D. Bykhovsky, and Y. Elovici, "Brightness: Leaking sensitive data from air-gapped workstations via screen brightness," in *Proc. 12th CMI Conf. Cybersecurity Privacy (CMI)*, 2019, pp. 1–6.

[26] M. Guri, B. Zadov, D. Bykhovsky, and Y. Elovici, "CTRL-ALT-LED: Leaking data from air-gapped computers via keyboard LEDs," in *Proc. Int. Comput. Softw. Appl. Conf.*, vol. 1, 2019, pp. 801–810.
[27] M. Guri, D. Bykhovsky, and Y. Elovici, "aIR-jumper: Covert air-gap exfiltration/infiltration via security cameras & infrared (IR)," 2017, *arXiv:1709.05742*.
[28] E. Carpentier, C. Thomasset, and J. Briffaut, "Bridging the gap: Data exfiltration in highly secured environments using Bluetooth IoTs," in *Proc. IEEE Int. Conf. Comput. Des. (ICCD)*, 2019, pp. 297–300.
[29] R. Ogen, O. Shwartz, K. Zvi, and Y. Oren, "Sensorless, permissionless information exfiltration with Wi-Fi micro-jamming," in *Proc. 12th USENIX Conf. Offensive Technol.*, 2018, p. 7.
[30] M. Tienteu et al., "Data exfiltration using building automation to bridge air gapped system," in *Proc. IEEE Int. Symp. Hardw. Orient. Security Trust (HOST)*, 2017. [Online]. Available: http://www.hostsymposium.org/host2017/hwdemo/HOST_2017_hwdemo_13.pdf
[31] S. Wendzel, B. Kahler, and T. Rist, "Covert channels and their prevention in building automation protocols: A prototype exemplified using BACnet," in *Proc. IEEE Int. Conf. Green Comput. Commun.*, 2012, pp. 731–736.
[32] "The legacy of KNX." KNX Association. 2020. [Online]. Available: https://www.knx.org/knx-en/for-professionals/What-is-KNX/KNX-History/index.php
[33] "BACnet Website." American Society of Heating, Refrigerating and Air-Conditioning Engineers. 2020. [Online]. Available: http://www.bacnet.org
[34] *Control Network Protocol Specification*, ANSI/CEA Standard 709.1-B-2002, 2002. [Online]. Available: https://webstore.ansi.org/Standards/CEA/ansicea7092002
[35] "EN ISO 16484—Building automation and control systems." EN ISO. 2016. [Online]. Available: https://ec.europa.eu/eip/ageing/standards/home/domotics-and-home-automation/en-iso-16484_en
[36] D. J. Brooks, M. Coole, P. Haskell-Dowland, M. Griffiths, and N. Lockhart (ASIS Found., Alexandria, VA, USA). *Building Automation & Control Systems: An Investigation into Vulnerabilities, Current Practice & Security Management Best Practice*. (2017). [Online]. Available: https://goo.gl/RM7ukP
[37] *KNX System Specifications—Architecture: Version 03.00.02*, KNX Assoc., Brussels, Belgium, 2009.
[38] "GoPacket project repository." Google Inc. 2022. [Online]. Available: https://github.com/google/gopacket
[39] V. Graveto, T. Cruz, and P. Simões, "A network intrusion detection system for building automation and control systems," *IEEE Access*, vol. 11, pp. 7968–7983, 2023. [Online]. Available: https://doi.org/10.1109/ACCESS.2023.3238874
[40] V. Graveto, L. Rosa, T. Cruz, and P. Simões, "A stealth monitoring mechanism for cyber-physical systems," *Int. J. Crit. Infrastruct. Prot.*, vol. 24, pp. 126–143, Mar. 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1874548218300672

Vitor Graveto received the B.Sc. degree, the master's degree in civil engineering, and the B.Sc. degree in informatics engineering from the University of Coimbra, Coimbra, Portugal, in 1989, 1999, and 2013, respectively, where he is currently pursuing the Ph.D. degree with the Department of Informatics Engineering.

His main research interests include areas, such as building automation and control systems, building management systems, cyber–physical systems security, and cybersecurity for critical infrastructures.

Tiago Cruz (Senior Member, IEEE) received the Ph.D. degree in informatics engineering from the University of Coimbra, Coimbra, Portugal, in 2012.

He has been an Assistant Professor with the Department of Informatics Engineering, University of Coimbra since December 2013. He is the author of more than 40 publications, including chapters in books, journal articles, and conference papers. His research interests include areas, such as management systems for communications infrastructures and services, critical infrastructure security, broadband access network device and service management, Internet of Things, software-defined networking, and network function virtualization.

Dr. Cruz is a member of the IEEE Communications Society.

Paulo Simões (Senior Member, IEEE) received the Doctoral degree from the University of Coimbra, Coimbra, Portugal, in 2002.

He is an Associate Professor with the University of Coimbra. He regularly leads technology transfer projects for industry partners, such as telecommunications operators and energy utilities. He has over 180 journal and conference publications in these areas. He has also been involved in several European research projects, with technical and management roles. His main research interests are security, network management, and critical infrastructure protection.