MSc. Cybersecurity Dissertation

Securing an OT Device: A Study of the Open Supervised Device Protocol

Peter Lionel Jones MBA CSyP FSyI MBCS
Student Number: 016186099

Supervisors: Dr. Philipp Reinecke, Eirini S Anthi

Industrial Sponsor: Third Millennium Systems Ltd.

Abstract

The adoption of the Open Supervised Device Protocol (OSDP) as an international standard (IEC 60839-11-5) offers access control systems manufacturers a standard bidirectional communication path to access control readers and other peripheral devices. OSDP has been designed to replace a technology called Wiegand that has been in use in commercial access control systems for the last 40 years. There are known vulnerabilities with Wiegand that make the technology prone to skimming and replay attacks. The secure channel feature of OSDP is intended to mitigate these vulnerabilities. In this dissertation, we consider if the OSDP protocol may be attacked in similar ways to Wiegand. We then develop a prototype to demonstrate how sensitive data may be leaked from two commercially available enterprise access control systems. We examine the implications of our findings and conclude by making a set of best-practice recommendations for installers and propose changes to the standard for consideration by the OSDP community.

Acknowledgements

For my late father, Lionel Kenneth Jones, whose dearest wish was for me to get my "letters".

For my son, Nathan Peter Jones, who one-day-soon will receive his "letters". May your dreams be big, your effort measured and your expectations overwhelmed.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

Large enterprises rely on sophisticated physical access control systems to determine which people may go where within an organisation, and at what time of day. An individual employee is provided with a credential, often embedded in a plastic card, which is presented to an access control reader to be read and verified against the access rights programmed into the system. With a focus on cybersecurity, organisations are now facing up to the reality that they need to upgrade the security of their systems because a variety of vulnerabilities that have been found in both the card and the reader technologies used to build these systems.

Over the last decade, key players in the industry have developed the Open Supervised Device Protocol, a bidirectional protocol incorporating a secure session using AES 128-bit encryption. OSDP Secure Channel is promoted as a way to secure the communication between an access control reader and a controller and addressing the key vulnerability of many legacy systems.

1.1 Motivation

The Open Supervised Device Protocol (OSDP) was adopted as international standard IEC60839-11-5:2020 (IEC, 2020) on 8th July, 2020. This protocol is now widely recommended by access control equipment vendors and security consultants as an open, interoperable and secure method of communicating between access control readers at the access points to an organisation and the

control equipment that make the decisions about granting entry.

A rich feature set is of no value, though, if the underlying protocol is not secure. Throughout, we critically evaluate the security features of OSDP. But to understand OSDP, we must first understand the capabilities and vulnerabilities of its predecessor: the Wiegand protocol. Wiegand has been the prevalent transmission protocol for access control readers for the last 40 years (Wehr, 2003) and, according to many industry experts, accounts for at least 80% of the installed base of access control systems. We will see how OSDP inherits some of the features unique to the Wiegand protocol.

The idea of building an OSDP analysis tool that could decrypt the protocol on-the-fly started in late 2020 when an email was received from Rodney Thayer. Rodney is a key technical player in the OSDP community. His email contained a log of activity from a Third Millennium reader for which he had some questions. The log file was in the **osdpcap** format (described in section 5.4.1.1), a JSON file format developed by Rodney as part of his work for SIA and the OSDP standardisation effort.

In the log file, it was observed that the initial key exchange was taking place prior to the communication switching to secure channel operation. This prompted the question: *"If the key exchange is in the log, can the log file be processed to show the encrypted data portion of the secure channel messages in plain text?"*

1.2 Aims and Objectives

In this dissertation, we seek to:

- Critically evaluate the Open Supervised Device Protocol from a cybersecurity perspective. How might we "skim" data from a live secure channel session?
- Propose a set of best practices for systems implementing OSDP. What do we need to do to assure the security of a system employing OSDP?
- Propose changes to the standard for consideration by the OSDP community.

The topics for the remainder of this dissertation are set out as follows:

In *"Chapter 2 - Literature Review"* we find no academic papers on OSDP and, instead, consider a related RS485-based protocol called MODBUS.

In *"Chapter 3 - Background"* we shall outline the structure of a modern access control system, examine the dominant communication method between readers and controllers, called Wiegand and provide an overview of OSDP.

In *"Chapter 4 - Research Methodology"* we outline the conceptual problems that need to be solved to build a prototype OSDP protocol analyser. We also define a test plan that allows us to apply the analyser to real-world systems.

In *"Chapter 5 - The Prototype"* , we distill the feature requirements for the OSDP analysis tool and describe the architecture and implementation of the prototype software we have called **osdpspy**. We highlight the design choices made in the implementation of the software, show examples of the problems the software solves for us and conclude by describing the installation and operation of **osdpspy**.

In *"Chapter 6 - Analysis"*, we evaluate the OSDP secure channel implementations of two commercially available access control systems. We detail the framework that we are going to apply to these systems, perform our analysis and then detail our findings for each of the systems. We show that **osdpspy** is a general purpose tool for security research into the OSDP protocol.

In *"Chapter 7 - Discussion"*, we consider the implications of our findings for OSDP and its practitioners. We summarise the vulnerability and how it might be exploited. We examine the role of the installer and the overall supply chain before looking at how the OSDP secure channel might be improved and how **osdpspy** can support education about the protocol and its practice.

In *"Chapter 8 - Conclusion"*, , we consider the implications of our work for the Open Supervised Device Protocol, its impact on embedded development at Third Millennium and the future of **osdpspy**. We also reflect up the time spent at Cardiff University.

Chapter 2

Literature Review

It is surprising to find that there are no academic papers on the Open Supervised Device Protocol. Instead, we look at the protocol that shaped the secure channel mechanism within OSDP and draw from the cryptanalysis of that protocol. We then turn our attention to another protocol used in a similar setting: MODBUS.

The OSDP secure channel is based on the Global Platform Secure Channel "03" (Global Platform Inc., 2014) a specification originally designed for contact smart cards running Java applets. Sabt and Traoré provide a cryptanalysis of this protocol (Sabt u.a., 2016) and argue that the protocol "probably satisfies the notions of strong security".

Even the smart card implementors have trouble in keeping up with advances in cryptography. Sabt and Traoré advocate the deprecation of SCP02 in favour of SCP03. This, perhaps, signposts what happens with many security schemes. They are only secure until they are not. We shall argue that this is also the case with the current secure channel implementation in OSDP.

The closest relation to OSDP from the perspective of application domain that we could find is MODBUS or, more specifically, MODBUS RTU. Devices are linked together on a bus with one controlling node make requests of the other nodes in turn.

The most startling thing about MODBUS RTU is that is has no security at all (Rinaldi, 2016). Indeed, if we examine the protocol specification (modbus.org, 2006), we find no mention of communication confidentiality.

We argue that related works on the MODBUS protocol are not relevant to an analysis of OSDP.

With no reference works or related material on OSDP or a protocol in a similar application domain with a security feature, we move to the next chapter, in which we outline the domain-specific knowledge required for this dissertation.

Chapter 3

Background

Enterprise access control systems are complex distributed systems typically comprising of components from multiple vendors. In this chapter we shall outline the structure of a modern access control system, examine the dominant communication method between readers and controllers, called Wiegand and provide an overview of OSDP and the state of adoption of this protocol by industry at the time of writing (June 2021).

3.1 The Enterprise Access Control System

There are a number of enterprise access control systems available on the market today. The OnGuard system from LenelS2 (LenelS2, 2020) and the C-CURE 9000 system from Software House (Software House, 2020) are examples of two systems used by governments and large enterprises around the world. The abstraction of the architecture described in Figure 3.1 is based on direct experience with these systems and maybe applied to the majority of access control systems on the market.

In the next few sections we provided definitions for each of the elements of the system, how these elements interact with each other and background information on their use in practice. We also consider the attack vectors in the operational technology domain that are of interest to us.

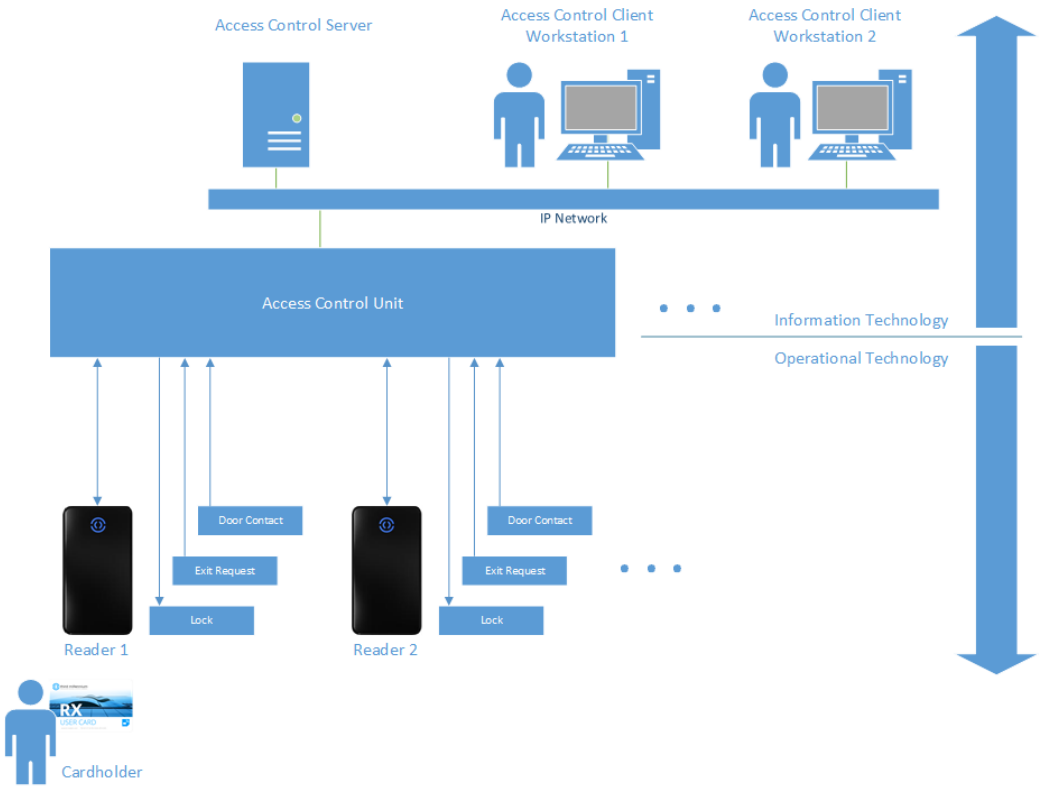

Figure 3.1: *Access Control System Architecture*

3.1.1 Cardholder

The cardholder may be an employee, a contractor, a visitor or in some other way related to the enterprise. As part of their relationship with the host organisation, they are issued with a credential that allows them to access certain areas. Access is gained by presenting the credential to an access control reader at the point of entry. The credential can take many forms: a magnetic stripe card, an RFID card or a Bluetooth credential stored on a mobile phone, for example. The enterprise will choose the credential type according to their needs. The enterprise may also enforce multi-factor user verification at the point of entry.

There is also a use case where the term cardholder is a misnomer. There are biometric identification technologies that can identify an individual from a fingerprint, an iris scan or some other physiological aspect of a human being, and turn this reading into a unique number compatible with the access control system to which they are connected.

The responsibility of the cardholder is to know when, where and how they may use the credential that has been issued to them.

3.1.2 Reader

The access control reader is an electronic device that reads a unique identifier for the cardholder from the organisation's credential of choice. There are many types of access control reader: from the single technology reader to a multi-factor biometric device.

Once the credential has been read, the reader may request a second factor (e.g. a biometric reader might require a fingerprint to be read) to verify the identity of the user.

Once this initial read phase has been completed, the unique identifier of the cardholder is transmitted to the access control unit (ACU). The reader may then be asked to display the result of the access control request by the ACU, or request a second factor (e.g. a keypad reader may request a PIN to be entered).

The responsibility of the reader is to read a unique identifier from a credential, or the cardholder in the case of biometric identification, and relay it to the ACU and to display the decisions of the access control unit.

3.1.3 Access Control Unit

The access control unit (ACU) provides the local decision making concerning whether a cardholder may gain access at a particular entry point. The ACU may support a number of readers and, therefore, entry points into and egress points out of the areas and/or buildings defined for the organisation.

When a unique identifier, or card number, is transmitted to the ACU by a reader, a decision must be made as to whether the cardholder should be granted access to the entry or egress point:

- Is this card number, known to the system?
- Is this card number allowed to access the specified entry or egress point?
- Is this card number allowed access at this time?

- Does the ACU require a second factor for this card number?

These are only some of the many decisions that can be made to determine access or otherwise for the card number read by the access control reader.

We should also note that there is not necessarily an equivalence between the card number and the cardholder. An access control system deals in card numbers and the ACU makes a decision on the card number delivered by the reader. How do we know that this number has come from a card presented by the cardholder?

In a basic form, the ACU is also responsible for monitoring the egress button used to unlock a door from the inside when there is no reader, the door monitor contact to monitor when the door is opened and closed and for firing the relay that unlocks the door.

In the modern access control system, the ACU sits on the enterprise's IT network. The ACU obtains a local database for local decision making from the access control server over this network. The database may either be pulled from the server by the ACU or pushed to the ACU by the server.

In Section 3.2 (*"Reader to Controller Communication"*), we will argue the communication between the reader and the ACU sits on an Operational Technology (OT) network. There are many potential threats that arise in this communication path that OSDP seeks to address that we will explore throughout this dissertation.

3.1.4 Access Control Server

The Access Control Server (server) is the "source of truth" for the entire system, maintaining the state of the system in the master database to be pushed towards the periphery (i.e. the Access Control Units and Readers) and receiving and storing events from these devices.

The Access Control Server runs a on one or more dedicated server machines that may be on-premise or in the cloud. All of the major systems vendors are now offering cloud-hosted solutions that are more convenient to configure through the use of virtual machines and, increasingly, container-based solutions.

Another mainstay of the modern Access Control Server is a Web API to allow in-

tegration with external services that can dynamically manage cardholders (e.g. a human resource management system).

3.1.5 Access Control Client

The Access Control Client (client) provides a user interface onto the overall system allowing an administrator to:

- Enter cardholders and their permissions onto the system.
- Configure the access control units (e.g. IP address, number of doors, readers).
- Configure reader operation (e.g. card only, card and PIN).
- Monitor events in real-time.
- Run reports on past events.
- Run reports against the cardholders configured in the system.
- Manage other administrators of the system.

The Access Control Client may be a desktop application (usually running on Windows) or a web browser consuming a web application hosted as a cloud service or implemented directly on the Access Control Server.

3.1.6 Attack Vectors

An access control reader has an input and an output. We concern ourselves with attack vectors on these two paths in this dissertation. The input is a credential using RFID or Bluetooth technologies, and the output is some form of communication to an access control unit. The principal objective of an attacker is to gain access through a door. This is achieved by spoofing the credential to the reader or by spoofing the data transmission from the card reader to the access control unit.

Whilst we do not present a spoofing attack on the OSDP communication between a reader and an access control unit in this work, we do demonstrate how

actionable intelligence may be gathered to facilitate a credential spoofing attack. The groundwork is, however, laid for a man-in-the-middle attack on the OSDP secure channel.

3.2 Reader to Controller Communication

As we shall soon see, OSDP exists because of a technology called "Wiegand". Wiegand is still used in the majority of access control systems operating today. OSDP seeks to address the shortcomings of Wiegand, but there are also concepts that directly transfer to an OSDP solution.

In this section, we explore the various aspects of Wiegand signposting the issues that led to the inception of OSDP. We also take a brief tour of other data transmission technologies that have been in common us in access control systems before providing an overview of the Open Supervised Device protocol itself.

3.2.1 Wiegand

The Wiegand transmission protocol is the most widely used method of communication between a reader and an access control unit. Wehr (Wehr, 2003) describes how the Wiegand effect was applied to a card technology. The readers for these cards had what became known as a Wiegand output.

As identified in an HID White Paper (HID Corporation, 2006), we should be careful with the term "Wiegand", though, as it has been used interchangeably over many years to mean:

- A specific reader-to-card interface. A "Wiegand Sensor".
- A specific binary reader-to-controller interface. A "Wiegand Interface".
- An electronic signal carrying data. A "Wiegand Output".
- The standard 26-bit binary card data format. The "Wiegand Format".
- An electromagnetic effect. The "Wiegand Effect" (Davis, 2011).
- A card technology. A "Wiegand Card".

We shall expand upon the definitions relevant to an access control reader.

3.2.1.1 Wiegand Card

A Wiegand card is constructed by laying strips of wire to form a binary code that will be read by a Wiegand-effect sensor to be transmitted to the control panel. Figure 3.2 (Mehl, 2018) shows an example of a Continental Instruments 36-bit formatted Wiegand strip. The wires laid in the upper half of the strip pass over the "zero" sensor to generate the zeros and the wires laid in the lower half of the strip pass over the "one" sensor to generate the ones in the output of the reader.

Figure 3.2: *A Wiegand Strip*

This strip was usually laminated into an ID card and swiped through the reader to generate the data stream to be transmitted to the access control unit using the Wiegand output.

3.2.1.2 Wiegand Output

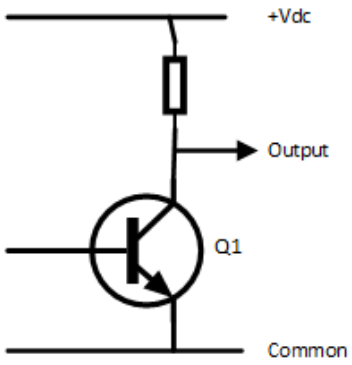

Figure 3.3: *Open Collector Output*

In the original Wiegand reader, manufactured by Sensor Engineering, the respective outputs are driven low by two open-collector outputs (shown in Figure 3.3) as the wires in the Wiegand strip are passed over the Wiegand-effect sensors in the read head. A resistor is shown in the circuit as it is common practice to but a weak pull-up in the output circuit of an access control reader to mitigate any interfacing issues with some equipment.

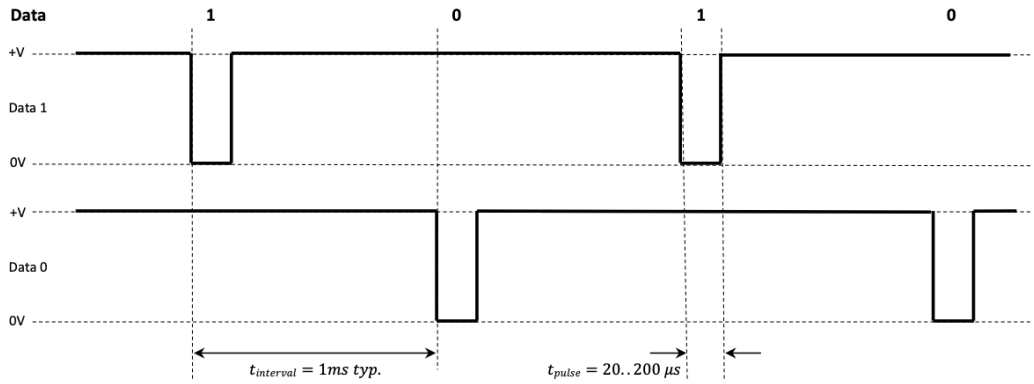

Figure 3.4: *Wiegand Protocol Timing*

Access control readers adopted this method in the 1980's to transmit bitstreams read from other card technologies such as, most notably, proximity cards. Unlike a real Wiegand reader, these "Wiegand emulation" readers used a fixed timing for the output as shown in Figure 3.4. Individual bits are typically spaced 1 millisecond apart with either the Data 0 (D0) or Data 1 (D1) outputs being driven low for between 50 and 100 microseconds.

3.2.1.3 Wiegand Interface

The term "Wiegand Interface" refers the input and output terminals of a Wiegand output reader that are used to connect the reader to an access control unit. The connections that are in common use are summarised in Table 3.1.

The access control unit will usually have a "Green" output that uses an open collector output to drive the corresponding Green input of the reader to illuminate the green LED when access has been granted. Additional connections (i.e. red, amber and buzzer) may be available between the reader and the access control unit.

Connection	Description
+V	Supply voltage, typically +5..12Vdc
0V	Ground or Common
D0	Data 0
D1	Data 1
GRN	Green LED input, signalled by the controller when entry has been granted.
RED	**[Optional]** Red LED input, signalled by the controller when access has been denied.
AMB	**[Optional]** Amber LED input, signalled by the controller to indicate that user input is required (e.g. to enter a PIN)
BUZ	**[Optional]** Buzzer input, signalled by the controller when an audible indication is required at the reader.

Table 3.1: *Wiegand Interface Connections*

3.2.1.4 Wiegand Format

Sensor Engineering and Wiegand also created what became a standard bit format for the transmission of card data: the Sensor 26-bit format, or more simply known as "26-Bit". This is an example of the term "Wiegand" used to describe "the standard 26-bit binary card data format".

This format is described in Figure 3.5. Cardholders are identified "uniquely" by a facility code and a card number. Some degree of integrity is provided to the format by the inclusion of two parity bits.

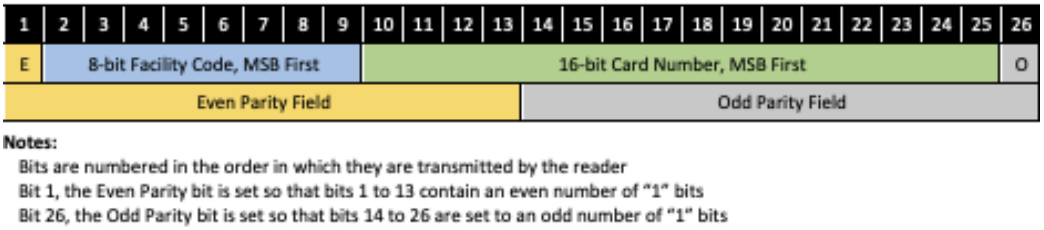

Figure 3.5: *Sensor 26-Bit Wiegand Format*

In Figure 3.6 we present an example of a Sensor 26-bit data frame programmed for facility code 99 and card number 1001.

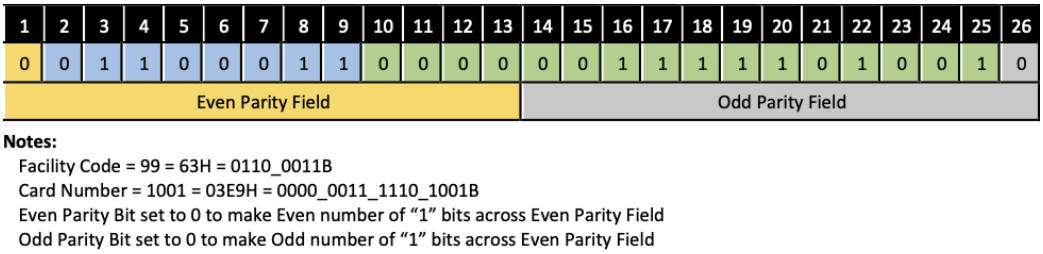

Figure 3.6: *Sensor 26-Bit Wiegand Format Example*

As the industry grew, the Sensor 26-bit format presented two principal problems:

- The range of facility codes and card numbers are limited.
- Credential vendors would, and still do, accept an order for cards with a specified facility code and card number without any check that the numbers have already been issued.

3.2.1.5 Proliferation of Formats

Vendors addressed the first problem by devising new formats to not only to provide larger number ranges, but also to lock their customers into their products. This approach hindered integration as vendors claimed security for their offering by keeping their proprietary formats secret.

The original Wiegand card technology was limited to a maximum of 37 wires and, therefore 37 bits. Initially, Wiegand formats were constrained by this 37-bit limit, but as the use of Wiegand cards waned, proximity and smart contactless readers (e.g. Mifare Classic, DESFire) adopted longer formats.

Third Millennium has acquired approaching 100 unique Wiegand formats over a period of 25 years that are widely used in industry.

3.2.1.6 Corporate 1000

In the late 90's, in an attempt to guarantee the uniqueness of the cards they supplied, HID Corporation (now HID Global) leveraged their market dominance to

introduce the Corporate-1000 format for their global customers. This program is still running today (HID Global, 2020). This approach potentially addressed both of the issues with the 26-bit format raised above.

A unique 12-bit facility code is provided to customers signing up for the programme. HID then manage the 20-bit card numbers issued to the customer based on a part number to guarantee the uniqueness of the cards supplied to the customer.

In addition to the managed facility codes and card numbers, HID also employed three parity bits covering interleaved sections of the data frame. This approach generally required additional work form the controller manufacturer to implement the scheme.

This approach served HID well until their proximity card technology could be cloned (a major vulnerability) and other card vendors started supplying HID-compatible cards including the Corporate 1000 format. For legal reasons, we are unable to publish the Corporate-1000 format here.

3.2.1.7 Limitations

Use of the Wiegand protocol only allows data to flow from the access control reader to the access control unit and beyond. This led to two use case scenarios where a reader based on the Wiegand protocol has noticeable limitations:

- Updating the reader's configuration
- Updating the reader's firmware

Scenario #1 - New Reader Configuration Required

Imagine that you are the security manager of a large corporation. You are responsible for an enterprise access control system with 1000 Wiegand output access control readers at its periphery. You have discovered that the credentials that you currently use have been compromised. You summon your access control vendors to meet with you and develop an action plan to mitigate the risk. You learn that your access control readers also support a newer more secure technology. The reader vendor can supply configuration cards to enable the new technology in your readers, as well as reading the existing cards until the entire card population has been replaced.

This scenario is very common and also very costly. Each reader in the system needs to be visited twice to **a)** enable the new technology whilst supporting the existing credentials and **b)** to disable the compromised technology once all of the cards have been replaced.

Scenario #2 - New Reader Firmware Required

In Scenario #1 you, as security manager, were fortunate that only a reconfiguration of the readers was required. In this second scenario, the reader vendor informs you that the reader hardware supports a newer, more secure credential technology and that the readers, which have a service port for just this occasion, will need a firmware upgrade to support this change to the system.

To perform a firmware upgrade, each reader in the system now needs to be visited by a skilled technician to remove the reader from the wall, make the firmware upgrade, reinstall and recommission the reader. What could be performed in a matter of seconds by one of your security staff now requires several minutes of contracted technician time per reader.

There has to be a better way.

3.2.1.8 Known Vulnerabilities

We can look at Wiegand through the lenses of confidentiality, integrity and availability:

Confidentiality - Wiegand was never designed for secure data transmission. All data is sent as plain text and can be easily skimmed, as we shall see below. The confidentiality of Wiegand is, therefore, non-existent.

Integrity - As we have seen in the construction of the Sensor 26-bit format, parity bits are added to the frame to to allow the receiver to detect bit transmission errors. This scheme is of minimal benefit as two changed bits within one of the parity fields will result in the reception of the wrong card number at the access control unit. We argue that this does not stand up to the integrity requirements of a modern information system. The integrity of Wiegand-formatted data is, therefore, minimal.

Availability - To deny service to the Wiegand transmission path, you need to remove the reader from the wall and interfere with the wiring. Shorting the data lines has the benefit of the reader appearing to work while sending jibberish

to the access control unit. Availability of Wiegand is, therefore, adequate. The assurance of availability can be improved by implementing and monitoring the tamper function if the reader supports it.

We now turn our attention to the two main vulnerabilities of Wiegand readers:

- Data may be easily skimmed.
- Most Wiegand readers are 125kHz proximity readers.

Skimming

We presume that an open collector output was originally chosen for its ability to drive data over a distance of 150m (the typical cable distance supported by Wiegand output readers) reliably and cheaply. The open collector outputs of a Wiegand reader also allow readers to be wired in parallel so that an access control unit can receive data from more than one reader on the same port. This opens the opportunity to skim the data transmitted and replay it to the access control unit at a later date.

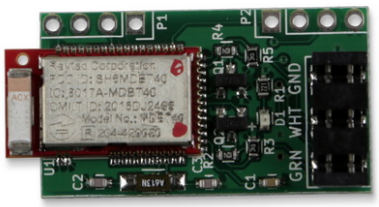

Figure 3.7: *BLE Key Wiegand Skimming Device*

For $35.00 we can buy a device known as a "BLE Key" (Hacker Warehouse, 2020) as shown in Figure 3.7. Installed covertly, this device captures and stores Wiegand frames to be replayed to the access control system using a mobile phone via the device's Bluetooth connectivity. This device was presented at the Blackhat USA Conference in 2015 (Baseggio / Evenchick, 2015).

Readers Using Proximity Technology

Whilst not a direct vulnerability of the Wiegand technology, a consequence of the success of HID Corporation (now HID Global, or simply HID) in promoting

the use of proximity technology for the best part of two decades is that most installed readers use or support some form of 125kHz proximity technology.

HID's 125kHz proximity technology was famously "hacked" by Seattle-based security research company IOActive in 2007. IOActive were due to present their findings but were stopped by HID threatening litigation (Help Net Security, 2007). HID had a reputation for always pursuing the legal option at this time.

Over time, more and more details of commercial 125kHz implementation came into the public domain and was published on proxmark.org. A summary of what was known in 2013 can be found on the website (proxmark.org, 2013). Coupled with experience and access to related materials, an engineering team team can quickly construct the equipment to read and create 125kHz proximity cards based on the information provided at proxmark.org. Third Millennium followed this route in 2013.

3.3 OSDP

3.3.1 History

The Open Supervised Device Protocol has its origins at Mercury Security in the early 2000s (OSDP Connect, 2016) and its early development was led by Frank Gasztonyi, President of the company. Mercury were one of the success stories of the industry as a developer and manufacturer of access control units that they sold to access control OEMs who delivered the complete system solution. Rather than mandated from the United States government, the protocol was developed by security equipment manufacturers in response to customer demands.

As a result of their success, Mercury were being asked to integrate a myriad of devices with their control panels and started to develop a standardised way of approaching the problem to share the load with the peripheral vendors leading to an initial specification in 2007. This version of the specification is commonly referred to as OSDP Version 1 and implemented common functions such as LED and buzzer control. It lacked, however, any form of secure channel between the control panel (or access control unit) and the peripheral device (a generic term that encompasses access control readers).

By 2009, a means to secure communication between the control panel and peripheral devices had been chosen. This heralded what the industry refers to as OSDP Version 2. This version of the specification was developed over the next five years by Mercury and its industry partners, including HID.

From the outset, Mercury recognised that to have lasting success, any specification that they developed with their industry partners would have to be handed to an industry body to steward, foster and promote the standard to access control vendors. In 2016, the OSDP specification was passed to the Security Industry Association or SIA (Security Industry Association, 2020).

One of the key contributions of SIA was establishing the OSDP Working Group which meets regularly to discuss various aspects of the OSDP standard including the proposal of new features. The objective of SIA was to get the OSDP adopted as an American (ANSI) or international standard (ISO/IEC). The endeavours of SIA and the OSDP Working Group led to this goal being achieved in 2020 with the adoption of OSDP Version 2.2. as IEC 60839-11-5:2020.

3.3.2 Benefits

In this section, we examine the key benefits of the OSDP protocol and compare and contrast with the features of Wiegand.

Secure Channel - Secure channel has been the main motivating factor for the adoption of OSDP over Wiegand by manufactures, integrators and end users alike. Wiegand transmits data as plain text. OSDP has the option of establishing a secure channel between the access control unit and the peripheral device. The security of the communication is based on an AES-128 session key and the application of CMAC chaining.

Constant Device Monitoring - Continuous monitoring of devices is possible with OSDP because the peripheral devices are constantly polled for events and status. This is where "Supervised" is used in the name of the protocol. If the device detects that it is under attack (i.e. in a tamper condition), it can be immediately reported to the access control unit.

Support New Device Technologies - The early 2000s saw advances in biometric and smart card technologies and the integration requirements for an access control system. Biometric devices and smart card readers are both supported by template management commands.

File Transfer - At one time, if a manufacturer-specific operation on a peripheral device was required, a series of manufacturer-specific commands had to be defined and sent to the peripheral device. While this is a viable approach for the peripheral device manufacturer, it is not practical for every access control unit manufacturer to implement support for a plethora of manufacturer-specific commands and use cases.

OSDP file transfer was devised to avoid this problem. An opaque file that has no meaning to the access control unit, is sent to a peripheral device. The peripheral device makes sure it is the intended recipient before processing the file.

3.3.3 Industry Adoption

OSDP is now widely specified as best practice for access control systems by security consultants. There has been a major educational push by consultants, equipment manufactures and the trade press about the vulnerabilities of Wiegand and the benefits of OSDP. This has led to a widespread adoption of at least the secure channel benefit of the protocol by both reader and access control unit manufacturers.

But with widespread adoption, manufacturers have been found to be implementing the protocol in subtly different ways that have led to interoperability problems. To address this concern, SIA have recently introduced the OSDP-Verified programme (Security Industry Association, 2020a). Access control equipment manufacturers submit their equipment for compliance and interoperability testing and, subject to passing these tests, the relevant products achieve OSDP Verified certification.

Third Millennium obtained an OSDP-Verified listing for its entire range of OSDP products in February 2021.

3.3.4 A Vehicle for Change

The introduction of OSDP to the enterprise access control system presents the opportunity to configure the reader devices from the centre of the system. This potentially changes the mental model of an enterprise access control system.

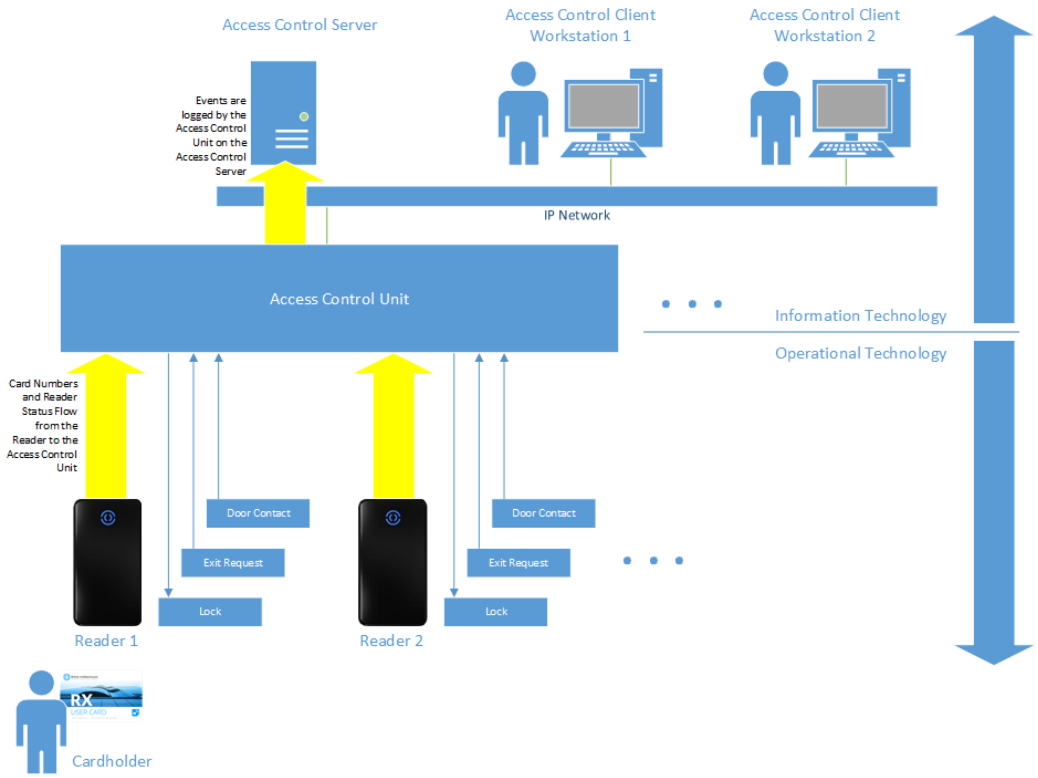

Figure 3.8: *Access Control System Architecture - Using Wiegand*

In a system with Wiegand readers, the data can only flow in one directions: card numbers flow from the reader to the access control unit for a decision and on to the access control server as an event as shown in Figure 3.8. This is the traditional mental model of the system.

With the introduction of OSDP supported by both the reader and the access control unit, data may now flow in both directions. As before, card numbers flow from the reader to the access control unit for a decision and then on to the access control server to be logged as an event. This is a minimum requirement. Data may also flow as a configuration, a firmware update or a simple command from the access control server to the access control unit and, finally, to the reader.

Even with this new capability, many practitioners retain the traditional mental model of a one-way flow from the reader. Most access control manufacturers have implemented secure channel to protect the data in transit from the reader but only a few have implemented features that can push data to the readers.

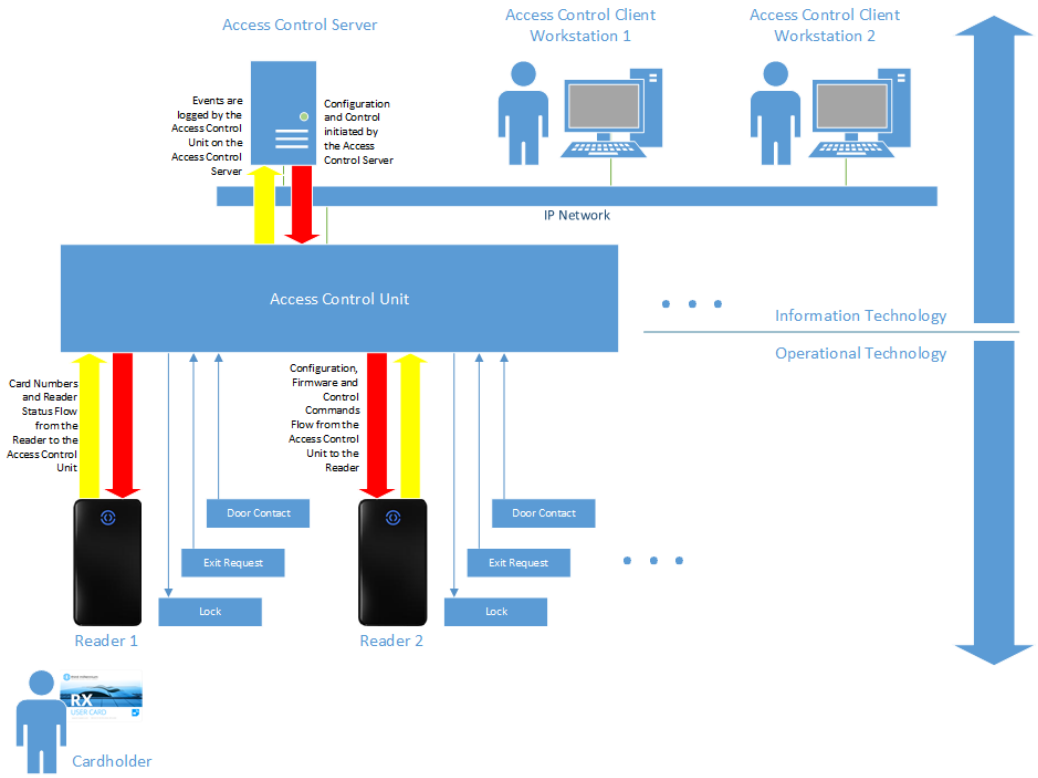

Figure 3.9: *Access Control System Architecture - Using OSDP*

3.4 IEC60839-11-5:2020

The Open Supervised Device Protocol (Security Industry Association, 2020) was recently adopted as international standard IEC60839-11-5:2020. In this section we give a brief overview of the features of the protocol and then introduce in more detail those features relevant to this dissertation:

- Data Transmission and Frame Structure
- Command Groups
- Secure Channel
- File Transfer

As with many formal specifications, in IEC60839-11-5:2020 the reader is taken quickly into the detail of the protocol that is being described. We argue that this

leaves a gap in understanding between the overview of the protocol and the detail of the implementation. In this section, and those that follow, we attempt to fill this gap by describing the supporting knowledge and the relevant features of the protocol in a more accessible form.

3.4.1 Data Transmission and Frame Structure

The Open Supervised Device Protocol uses a half-duplex RS-485 communication bus as the physical layer, prescribing an asynchronous serial format and a range of baud rates.

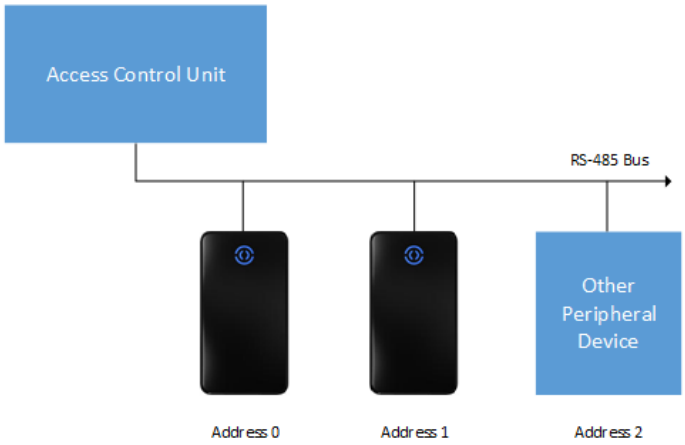

Figure 3.10: *OSDP Device Bus*

The master of the bus is the access control unit (ACU) which communicates with one or more peripheral devices (PD) as shown in Figure 3.10. Peripheral devices are typically access control readers but the standard features of the protocol support input and output so that door control and other security-related devices may also be developed.

The ACU sends a command to an addressed PD using the structure shown in Figure 3.11. The PD processes the command encapsulated in the frame and responds to the ACU using the same frame structure.

Additional commentary for each of these fields is povided below:

SOM - Signifies the Start-Of-Message and always has the value 53H.

ADDR - The most significant bit indicates f this message is a command (0) from

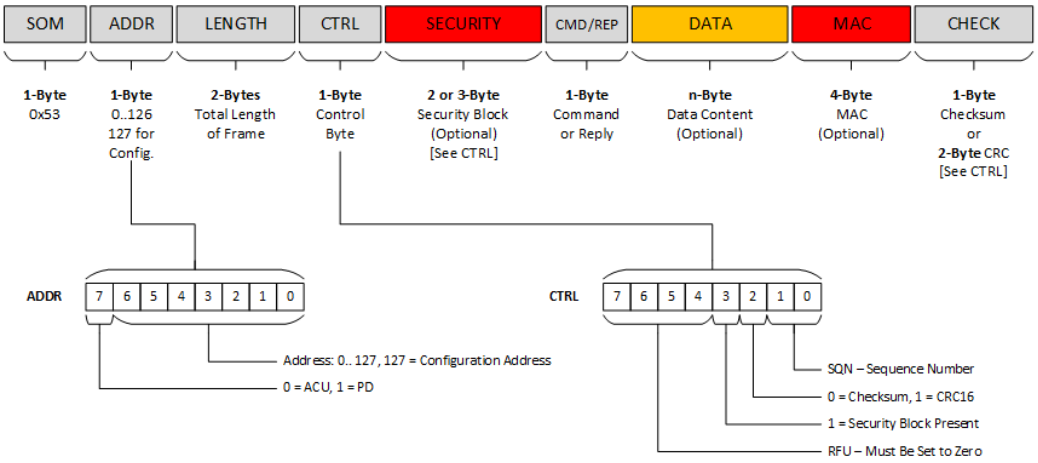

Figure 3.11: *OSDP Frame Structure*

the ACU or a response (1) from the PD. The 7 least significant bits contain the address of the corresponding PD from 0 to 126. Address 127 is the special configuration address.

LENGTH - The overall length of the frame as a number of bytes.

CTRL - The Control byte provides information about the remainder of the frame including the method type used verify the integrity of the frame transmission. The Control byte also includes a sequence number so that the devices engaged in communication may identify out of sequence frames.

SECURITY - The Security Block optionally contains commands and responses for the secure channel protocol. The Security Block is present of the appropriate bit has been set in the CTRL byte.

CMD/REP - The Command to be executed by the PD or the Reply from the PD as idenified by the most significant bit of the ADDR field.

DATA - The Data field is only present of the command or reply contains data. If the secure channel has been established and data is present, the data will have been encrypted using the session key.

MAC If this is a command or reply executed after the secure channel has been established, a 4-byte MAC will have been added to the frame to further affirm the integrity of the frame.

CHECK - A 1-byte Checksum or a 2-byte CRC16 added to the frame to verify the

integrity of the transmission.

3.4.2 Command Groupings

In Table 3.2, we classify OSDP commands according to their function. This provides an overview of the likely capabilities of a Peripheral Device.

All compliant peripheral devices implement the **Identification**, **Configuration** and **Secure Channel** function groups. Additional commands are added to the vocabulary of a peripheral device depending on the required capabilities.

In the next two sections we shall explore the functions of the **Secure Channel** and **File Transfer** function groups.

Group	Commands	Comments
Biometric	osdp_BIOREAD osdp_BIOMATCH	Provides the control functions to communicate with a biometric reader.
Configuration	osdp_COMSET, osdp_ACURXSIZE	Used to configure the address and baud rate of the peripheral device and to advise the peripheral device of the maximum receive buffer of the access control unit.
Event Reporting	osdp_POLL	Polls the peripheral device for events and changes of status. For an access control reader, card reads are reported in response to this command.
File Transfer	osdp_FILETRANSFER	Used to transfer data to a PD in an opaque manner.
Identification	osdp_ID osdp_CAP	Used to identify the peripheral device and its capabilities.
Output Control	osdp_BUZ osdp_LED osdp_OUT osdp_TEXT	Allows the ACU to control the output functions of the peripheral device including the LED, the buzzer, output and text functions if they are supported.
Secure Channel	osdp_KEYSET osdp_CHLNG osdp_SCRYPT	Used to configure the secure channel and and also to share the secret key.
Smart Card	osdp_XWR osdp_ABORT osdp_PIVDATA, osdp_GENAUTH osdp_CRAUTH osdp_KEEPACTIVE	Provides a set of commands to interface with a smart card reader.
Status Reports	osdp_LSTAT osdp_ISTAT osdp_OSTAT osdp_RSTAT	Allows the ACU to solicit the status of various functions of a peripheral device.

Table 3.2: *OSDP Command Groups*

3.4.3 Secure Channel

The Secure Channel is the basis for secure data transmission in OSDP and also the main reason for the adoption of the protocol by security equipment manufacturers. In this section we shall show how the Secure Channel session is established between an access control unit and a reader.

3.4.3.1 Establishing the Secure Channel

Secure communication between the access control unit and the peripheral device relies on a set of session keys being established for the two devices. This key is generated by computation based on a pre-shared key and an exchange of random numbers. An additional integrity check is provided by the implementation of a rolling MAC included in each frame as shown in Figure 3.11.

Figure 3.12 shows the flow of data between the access control unit and the peripheral device. You will see that each step of the process uses a different Security Block Type denoted by the **SCS** prefix. For each of these steps there is a corresponding command or reply defined in the OSDP specification.

Before the exchange can start, the access control unit and the peripheral device must have a pre-shared AES 128-bit key called the Secure Channel Base Key or SCBK. Each peripheral device typically has a default SCBK known as the SCBK_D. This default key is necessary to bootstrap the setting of a system specific key for the device because the key can only be set in secure channel.

Once both the access control unit and peripheral device are sharing the same SCBK, the four-tesp process to establish the secure channel may begin.

STEP 1: SCS_11, ACU→PD

The access control unit sends a 64-bit random number $\mathrm{RND_A}$ to the peripheral device as an array of 8 bytes as a parameters in the osdp_CHLNG command.

The peripheral device receives the random number $\mathrm{RND_A}$.

Both devices generate the session key set as follows:

$$S\_ENC = ENC([0x01, 0x82, RND_{\mathrm{A}}[0]..RND_{\mathrm{A}}[5], 0x00..0x00], SCBK)$$

$$S\_MAC1 = ENC([0x01, 0x01, RND_{\mathrm{A}}[0]..RND_{\mathrm{A}}[5], 0x00..0x00], SCBK)$$

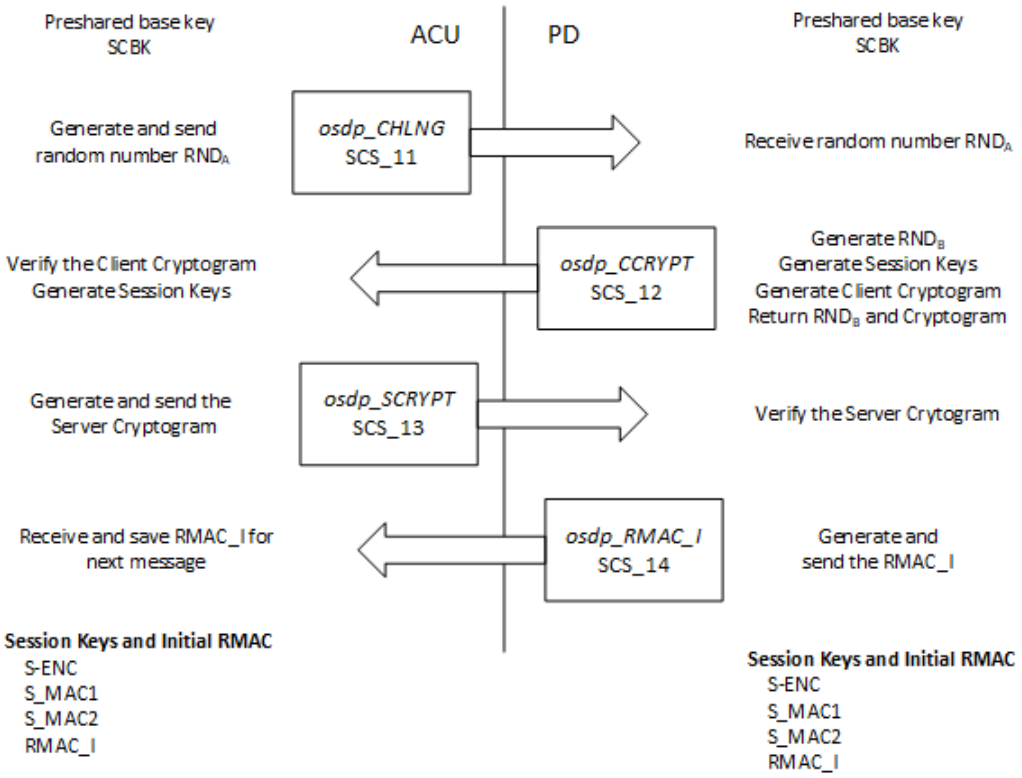

Figure 3.12: *Establishing the Secure Channel*

$$S\_MAC2 = ENC([0x01, 0x02, RND_{\mathrm{A}}[0]..RND_{\mathrm{A}}[5], 0x00..0x00], SCBK)$$

Where:

ENC is an encryption function with two parameters.

The first parameter is the 16 byte array to be encrypted.

The second parameter is the AES128 key to be used to encrypt the data.

S_ENC is a session key used to encrypt the data field of the frame.

S_MAC1 is a session key used in calculating the MAC of the frame.

S_MAC2 is a session key used in calculating the MAC of the frame.

The peripheral device generates a 64-bit random number RND_B and then the client and server cryptograms as follows:

CCRYPT = ENC([RND_A[0]..RND_A[7], RND_B[0]..RND_B[7]], S_ENC)

SCRYPT = ENC([RND_B[0]..RND_B[7], RND_A[0]..RND_A[7]], S_ENC)

Where:

CCRYPT is the client cryptogram.

SCRYPT is the server cryptogram.

The cryptograms are used to prove that both the access control unit and the peripheral device are using the same shared SCBK.

STEP 2: SCS_12, PD→ACU

The peripheral device sends the Client UID, RND_B and CCRYPT as an osdp_CCRYPT response to the access control unit.

The Client UID is a unique number used to identify the peripheral device and is similar to a MAC address. This number may be used by the access control unit to generate the SCBK for the peripheral device.

The access control unit receives and verifies the CCRYPT.

STEP 3: SCS_13, ACU→PD

Now that the access control unit has both RND_A and RND_B, it can generate the server cryptogram SCRYPT in the same way as the peripheral device. This SCRYPT, generated by the access control unit, is then sent to the peripheral device in an osdp_SCRYPT command.

The peripheral receives the SCRYPT computed by the access control unit and compares it to the value computed earlier.

STEP 4: SCS_14, PD→ACU

With the SCRYPT values matching, the peripheral device generates the RMAC_I as follows:

x = ENC(SCRYPT, S_MAC1)

RMAC_I = ENC(x, S_MAC2)

Where:

x is an intermediate 16-byte buffer.

RMAC_I is the initial vector for the first MAC calculation.

The peripheral device sends the RMAC_I to the access control unit in an osdp_RMAC_I response to synchronise the devices for secure communication.

The RMAC_I is the initial vector for for calculating the MAC (message authentication code) for the first of the secure frames sent by the access control unit to the peripheral device. Whilst the current MAC is maintained as a full 16-byte vector, for the sake of efficiency, only the first four bytes of the MAC appear in a message.

At this point, the secure channel has been established between the access control unit and the peripheral device and secure messaging may begin.

3.4.3.2 Maintaining the Secure Channel

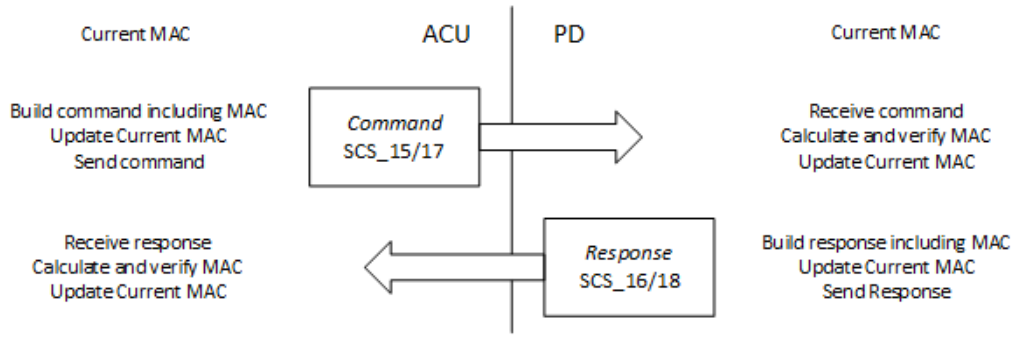

Figure 3.13: *Maintaining the Secure Channel by Rolling the MAC*

As secure messages are sent between the devices, the MAC is maintained and rolled by both the access control unit and the peripheral device to maintain the synchronisation between the two devices.

As shown in Figure 3.13, the MAC is calculated and updated as new frames are sent and received. We can consider this as a two step process to deal with case of command from the access control unit and a response from the peripheral device.

STEP 1: ACU→PD

The current MAC value is for the for both the access control unit and the peripheral device. The access control unit prepares the next command to be sent to the peripheral device.

Before the command can be transmitted, the access control unit must calculate the the MAC fro the frame as follows:

$$MAC_{\text{NEW}} = f(CMD, MAC_{\text{CUR}}, S\_MAC1, S\_MAC2)$$

Where:

f is the function that generates the new MAC.

CMD is the OSDP command frame to be sent to the PD.

MAC_{CUR} is the current MAC value stored in the ACU.

MAC_{NEW} is the new MAC that has been calculated

S_MAC1 is a session key generated when the link was established

S_MAC2 is a session key generated when the link was established

The current MAC value is updated with the new 16-byte MAC vector that was just calculated. The first four bytes are added to the frame followed by the check code which must be a CRC16 for a secure channel message.

The fully-constructed command is then send to the peripheral device which repeats the MAC calculation based on the frame that it has just received. The first four bytes of the newly calculated MAC should match the MAC field of the incoming frame, in which case the current MAC of the peripheral device is updated.

STEP 2: PD→ACU

The peripheral device processes the command it has received from the access control unit and prepares the response. Before the response can be sent, the perpheral device must calculate the MAC for the response using exactly the same mechanism as the access control unit used in **STEP 1**.

The current MAC value is updated with the new 16-byte MAC vector that was just calculated. The first four bytes are added to the response followed by the CRC16 check code. The response is then sent to the access control unit.

The access control unit processes the response it has received from the peripheral device and starts preparing the next command and thereby continuing the cycle.

3.5 Summary

In this chapter, we have laid out a summary of the domain-specific knowledge needed by a reader of this dissertation. We have learned about the various elements of an enterprise access control system and how they work together. We have examined the evolution and highlighted the vulnerabilities of Wiegand, the dominant transmission protocol for access control readers for the last 40 years. We have also explored the background to Wiegand's successor-designate: OSDP. We have learned how OSDP was born of necessity, how it grew to meets the needs of its stakeholders and how, finally, it became an industry standard with a mature compliance programme.

Chapter 4

Research Methodology

4.1 Introduction

The core assertion of this dissertation is that secure channel communication between an access control unit and a reader may be "skimmed". By skimmed, we mean that the protocol may be decrypted and the plain text information extracted for further processing and interpretation. This data may be used to verify a secure channel implementation by an engineering team or, alternatively, it could be used as a component of an attack on a system by a bad actor.

In this chapter, we break down our research into the following discrete steps :

- Observe the Protocol
- Witness the Key Exchange
- Decrypt the Protocol
- Analyse the Protocol in Real-Time
- Build a Prototype Protocol Analyser
- Analyse Real-World Systems

4.2 Observe the Protocol

Before we can analyse the OSDP protocol, we must first capture the frames exchanged between the access control unit and the reader or other peripheral device. This data may be obtained from one of two sources:

- An **osdpcap** log file.
- A live RS-485 bus.

4.2.1 Analysing an osdpcap Log File

In section 1.1, we described how a **osdpcap** log file provided the initial idea for decoding and decrypting the OSDP protocol. A log file provides reproducible material for the initial work of breaking down the encryption process: the software to solve this particular problem can be tested repeatedly against the same input to verify operation.

4.2.2 Analysing a Live OSDP Bus

The OSDP protocol typically runs on an RS-485 multi-drop bus. Activity on the bus may be easily monitored using an RS-485 USB dongle, providing a stream of bytes to be decoded in real-time by software. Successfully monitoring, decrypting and decoding the OSDP protocol in real-time would provide a useful tool to system designers and hackers alike.

Of course, any tool that can perform live capture of the OSDP protocol should also have the capability to output **osdpcap** log files. The original **osdpcap** file was generated by an open-source compliance tool provided by SIA.

4.3 Witness the Key Exchange

4.3.1 Obtaining the Session Keys

In Figure 4.1 we, once again, show how the secure channel is established.

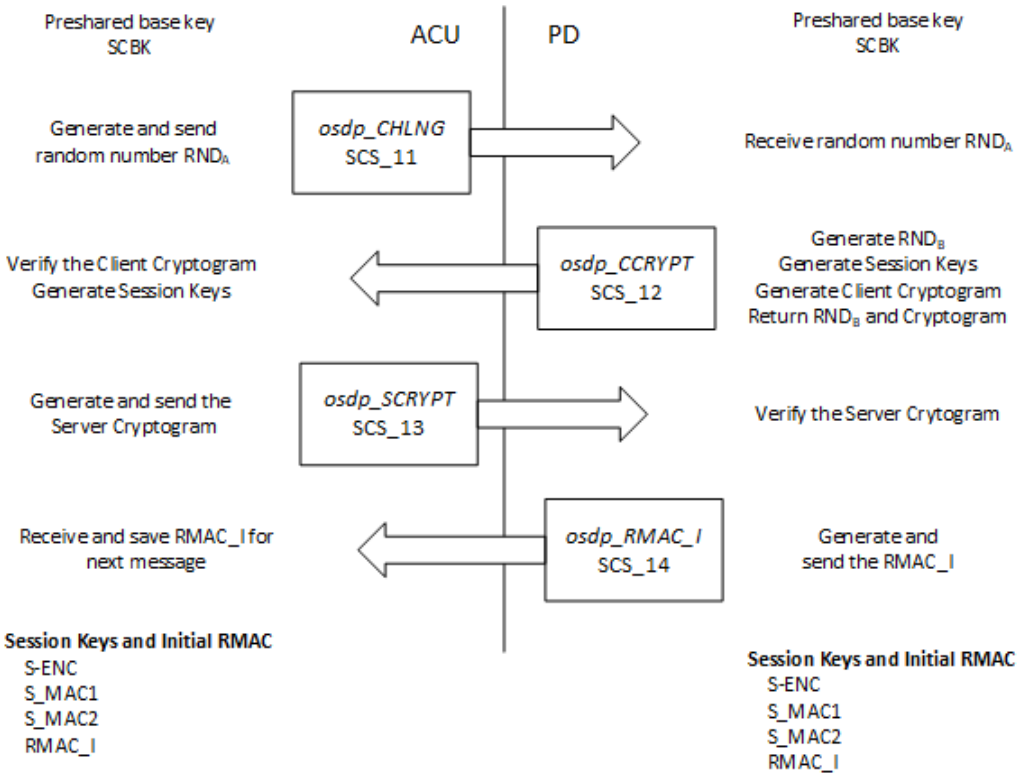

Figure 4.1: *Establishing the Secure Channel*

A feature of the **osdp_CHLNG** command is the selection of the secure channel base key to be used to start the session. This simplifies the detection of the use of the default secure channel base key (referred to as **SCBK-D** in the specification) to set up the secure channel.

In Figure 4.2 we show how the secure channel exchange can be monitored and the session keys generated by observation. The secure channel base key (**SCBK**) used can be verified against the client and server cryptograms to confirm that the session keys have been generated correctly.

Once the secure channel setup has been observed and the **S_ENC**, **S_MAC1**, **S_MAC2** and **RMAC_I** values calculated and verified, we can decrypt the protocol frame-by-frame as the access control unit communicates with the reader.

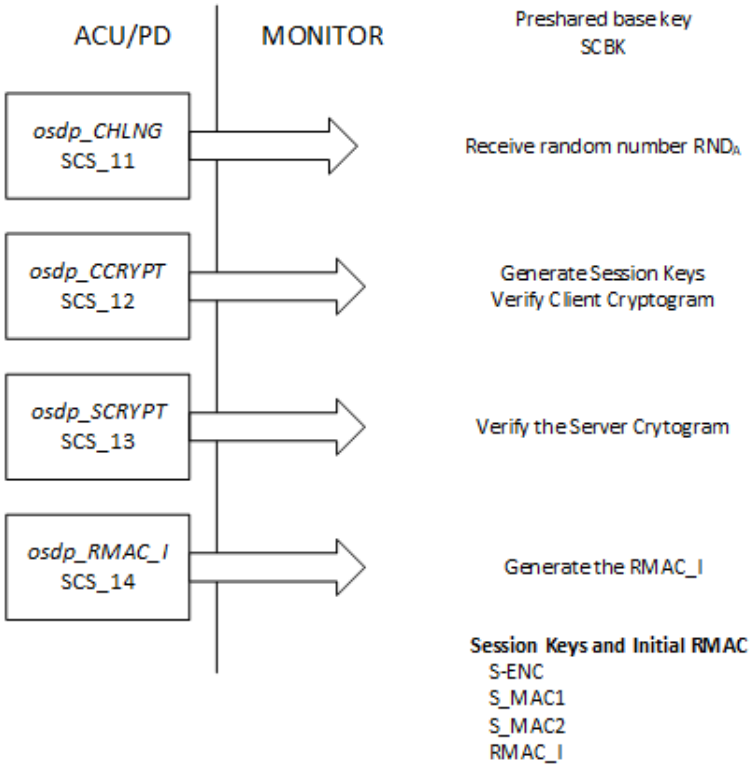

Figure 4.2: *Monitoring the Secure Channel*

4.3.2 Key Exchange

The OSDP protocol requires that the **SCBK** is set in a secure session using the **osdp_KEYSET** command. This is normally done as part a broader secure channel setup process which consists of the following steps:

1. Set the access control unit and reader into "pairing" mode.
2. Establish a secure channel session using the **SCBK-D**.
3. Set the new **SCBK** using the **osdp_KEYSET** command.
4. Establish a secure channel session using the new **SCBK**.

The challenge for the prototype protocol analyser is to reliably follow this sequence and to remain synchronised with the current secure session between the access control unit and the one or more readers connected to it.

4.4 Decrypt the Protocol

Once the decoding software is synchronised with the secure session, it is possible to decrypt the payload from each command and response where a payload is present. It is from these payloads that sensitive data may be leaked from the protocol.

From decrypted protocol activity, it should be possible to gather the following information:

- Secure Channel Base Key(**SCBK**)
- Card Numbers
- PIN Numbers
- Valid Access
- OSDP Files

Acquisition of this data is the primary objective of this dissertation.

4.4.1 Secure Channel Base Key

Once the the secure channel can encrypted, the first piece of information that will be leaked is from the payload of the **osdp_KEYSET** command is the new Secure Channel Base Key. This payload contains the new **SCBK**, an AES-128 encryption key.

This data may also be persisted for later use when the ACU attempts to start a secure session directly with an **SCBK** that has been previously set.

4.4.2 Card Numbers

Card numbers are the unique identifiers used by an access control system that correspond to the actual users of a system. This data may be obtained from the **osdp_RAW** and **osdp_FMT** responses to the **osdp_POLL** command.

The **osdp_RAW** response may contain a known Wiegand format as described in Section 3.2.1.4 onwards. The decoded form of this data might be useful to an attacker.

4.4.3 PIN Numbers

Some doors on a system may require the entry of a PIN. Each key press on an OSDP access control reader will return an **osdp_KEYPAD** response to the **osdp_POLL** command. By recording the key presses, it may be possible to assemble the PIN numbers corresponding to each of the cards used at a reader.

4.4.4 Valid Access

When entry is granted at a door, the LED of the reader is usually turned Green by the access control unit sending an **osdp_LED** command. By monitoring for this command, it is possible to trigger the analysis of the card number and the PIN codes entered into the reader to identify a card and the corresponding PIN number required to gain access to the door.

This is an example of accumulating data over time towards a goal.

4.4.5 OSDP Files

Version 2.2 of the OSDP specification introduced file transfer to the protocol. File transfer allows device-specific information to be transferred from the access control unit to the reader. The protocol provides the **osdp_FILETRANSFER** command to achieve this.

Files are split up into fragments and sent to the reader piece-by-piece. By observing the file transfer, we assert that it is possible to assemble the file and save it to disk for later processing.

4.5 Build a Prototype Protocol Analyser

In order to evaluate the ideas presented earlier in this chapter, we propose to build a software prototype that implements the various methods to decrypt the protocol and achieves the payload recovery objectives.

This chapter forms the basis for the feature requirements outlined in the next chapter.

4.6 Analyse Real-World Systems

Once we have developed of the prototype OSDP protocol analyser, we shall apply it to two commercially available access controls systems.

In each case, we shall execute the following test plan:

1. Describe the system and the process by which secure channel communication is initiated.
2. Detect the key exchange and report the key set by the access control unit.
3. Capture and report encrypted card data.
4. Capture and report PIN entry.
5. Capture card numbers, complete PINs and report triggered by an **osdp_LED** command.
6. Capture and save any files transferred from the access control unit to reader using the **osdp_FILETRANSFER** command.
7. Summarise findings of the tests.

4.7 Conclusion

In this chapter, we have outlined the conceptual problems that need to be solved to build a prototype OSDP protocol analyser. We have also defined a test plan that allows us to apply the analyser to two real-world systems.

Chapter 5

The Prototype

5.1 Introduction

In this chapter, we distill the feature requirements for the OSDP analysis tool and describe the architecture and implementation of the prototype software we have called **osdpspy**. We highlight the design choices made in the implementation of the software, show examples of the problems the software solves for us and conclude by describing the installation and operation of **osdpspy**.

5.2 Language and Framework Selection

Third Millennium has been developing desktop utility applications to support OSDP development and other functions in C# and .NET since 2016. Given the availability of related .NET libraries to support this development already existed, this was reason enough to use one of the .NET frameworks and C# for this project. Any reusable code generated during this development could be fed back into the Third Millennium code base.

In this section, we consider other programming languages that may have been used to implement the prototype before introducing two .NET libraries that will be used to provide input and output functions for the prototype.

5.2.1 Other Languages

Other languages that could have been considered had the argument for reuse not been so compelling included:

- **Java** - Java and C# are similar languages. Indeed, Third Millennium has produced software components in Java for the Android operating system and the language has good cross-platform support. Java also has excellent third-party library support through repositories such as the MVN Repository (MVN Repository, 2021).
- **Python** - Python would have been an interesting learning opportunity for this project. There is strong third-party library support through Python Package Index (Python Software Foundation, 2021).
- **C/C++** - Neither of these languages were chosen for the application because of the extra libraries that would have needed to be developed for structured logging integration and cross-platform support. There have been a number of language features added to C++ in recent years which would also have made this an interesting learning opportunity.

5.2.2 Structured Logging

It is now standard practice at Third Millennium to follow SOLID principles in software design. In order to support the Inversion-of-Control principle, we have used the standard Microsoft Dependency Injection framework. This framework comes with a standardised logging output through the *ILogger* interface.

The .NET Serilog library (Serilog, 2021) is available as a NuGet package and provides an *ILogger* implementation that supports structured logging. Structured logging allows **osdpspy** to push data to an event sink as a set of timestamped key/value pairs. The Serilog library serves as an intermediary to numerous commercial event sink products including:

- Elasticsearch (Elasticsearch, 2021)
- Seq (Datalust, 2021)
- Splunk (Splunk, 2021)

By logging the the data produced by **osdpspy** to one of these event sinks, we can use the power of these tools to query this data both retrospectively and also in real-time. We will, therefore, specify structured logging as one of the the outputs from the **osdpspy** tool.

5.2.3 Command Line Parsing

The simplest way to prototype the OSDP analysis tool is as a command line tool because the application will not have the overhead of providing a graphical user interface. In order to handle the configuration of the prototype, we will use command line parameters. This presents the problem of parsing a command line to input the configuration into the application.

We will use the .NET **McMaster.Extensions.CommandLineUtils** library (McMaster, 2021) so that we can easily build complex command lines such as:

osdpspy listen -p /dev/tty.usbserial-AQ00SUTK -s http://my-seq-logger

In this example, we are asking the prototype to sample data in real-time from the **/dev/tty.usbserial-AQ00SUTK** and log the output to a Seq event sink at **http://my-seq-logger**.

The library allows command line parsing to be easily integrated into the application.

The additional benefit of developing a .NET command line application is that it can be built for Windows, MacOS and Linux allowing researtches to use the prototype on their platform of choice.

We will, therefore, specify a command line interface as the user interface for **osdpspy** tool.

5.3 Feature Requirements

Based on the research methodology outlined in the previous chapter and the selected support libraries, we can define the feature requirements of the OSDP analysis tool as follows:

1. The tool shall be a command line utility capable of running on Windows 10, MacOS or Linux.
2. The tool shall have the capability to consume and produce OSDP log files as defined by SIA. (Security Industry Association, 2021)
3. The tool shall be able to decode OSDP frames and break out the various elements of each command and response.
4. Subject to the log containing a pairing of a controller with an access control unit, the tool shall be able to decrypt secure channel communication.
5. The tool shall have the capability to receive and process OSDP protocol frames in real time by attaching to an OSDP bus.
6. The tool shall have the capability to log OSDP activity in real time to a structured event sink such as Seq, Elasticsearch, Splunk et al.
7. The tool shall have the capability to capture and store files passed from the ACU to the reader using **osdp_FILETRANSFER** commands.

This feature requirement list was designed as a sequential set of problems, progressively enhancing the program as each of the features was completed. The intent is to produce a useful debugging and security research tool for the OSDP protocol.

5.4 Software Architecture

In Figure 5.1, we present the software architecture of the prototype. In this section we describe the inputs to, outputs from and the constituent components of the application, as shown in the architecture diagram.

5.4.1 Inputs

The input to the application may come from either an **osdpcap** trace file or live from an RS-485 bus.

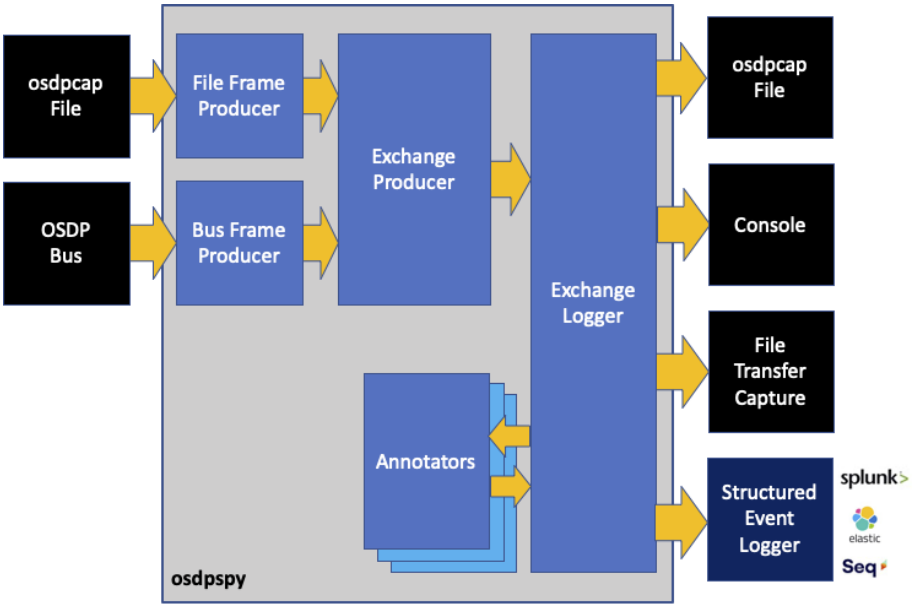

Figure 5.1: *Software Architecture*

5.4.1.1 osdpcap File

The **osdpcap** file format is a standard method by which traces of OSDP communication may be shared by interested parties. The initial log files sent as part of an on-going conversation about Third Millennium readers were in a form close to the current specification but had not followed a convention for field-naming that would be readily portable. It was suggested that a camel-case naming convention be used and the current file format was created.

An example of two entries from an OSDP trace is shown in Listing 5.1. The trace contains an osdp_POLL command from the access control unit followed by an osdp_ACK response from the peripheral device. For the purposes of the prototype, the combination of a command from the access control unit with the corresponding response from the peripheral device is called an **exchange**.

There are two pieces of information that are of interest to the application: the raw frame data in the **data** field and the timestamp for the frame in the **timeSec** and **timeNano**.

An **osdpcap** log file may be processed by using the **import** subcommand with the file specified by the **-i** option:

```
{       "timeSec" : "1613258466",
        "timeNano" : "529474046",
        "io" : "trace",
        "data" : " ff 53 00 08 00 05 60 da 99",
        "osdpTraceVersion":"1",
        "osdpSource":"libosdp-conformance 0.91-1" }

{       "timeSec" : "1613258466",
        "timeNano" : "604015953",
        "io" : "trace",
        "data" : " 53 80 08 00 05 40 68 9f",
        "osdpTraceVersion":"1",
        "osdpSource":"libosdp-conformance 0.91-1" }
```

Listing 5.1: OSDPCAP Example

osdpspy import -i logFile.osdpcap *[other options]*

The **import** subcommand uses the **File Frame Producer** component to assemble the frames for processing.

5.4.1.2 OSDP Bus

Whilst working retrospectively with **osdpcap** log files is of some value, working in real-time with a live OSDP communication bus delivers real benefits to the security researcher and attack alike.

A USB serial port is attached to the computer to provide a live stream of data into the application to be assembled into protocol frames and exchanges.

The serial port must be specified using the **-p** option when the **listen** subcommand is used, as in the following example:

osdpspy listen -p /dev/tty/usbserial-XX00ZZZZ *[other options]*

The **listen** subcommand uses the **Bus Frame Producer** component to assemble the frames for processing.

5.4.2 Application Components

The application components essential form an information factory. The raw material for this factory are the inputs we have defined in Section 5.4.1. This raw data is processed by the application components into information useful to a security researcher or attacker as define in Section 5.4.3, Outputs.

5.4.2.1 File Frame Producer

The **File Frame Producer** component is responsible for reading data in from an **osdpcap** log file configured by the **import** subcommand of the prototype. The data from the file is assembled into **Frame Products** for consumption by the **Exchange Producer** component.

A **Frame Product** contains the following information:

- A timestamp of when the frame was transmitted or received by the access control unit.
- The raw OSDP frame data as a byte array.
- The payload of the OSDP frame in both plain text and cipher text as available.

5.4.2.2 Bus Frame Producer

The **Bus Frame Producer**, like the **File Frame Producer**, is responsible for making **Frame Products**. In this case the component opens a serial port configured by the application command line and monitors the RS-485 communication bus on which the OSDP protocol is running. The component syncronises with the data stream and assembles and timestamps **Frame Products** for consumption by the **Exchange Producer**.

5.4.2.3 Exchange Producer

An **Exchange** is the basic unit of analysis used by the prototype OSDP analysis tool and contains the following information:

- An auto-incremented sequence number to uniquely identify the **Exchange**.
- A **Frame Product** from the Access Control Unit.
- A **Frame Product** from the Peripheral Device, if present.

This unit of analysis was chosen because all communication in OSDP is point-to-point and the intent is for the access control unit to always receive a response from the peripheral device. This does not always happen and will be evident from a missing **Frame Product** for the peripheral device.

The responsibility of the **Exchange Producer** component is to consume **Frame Products** and pair them up into **Exchanges** between the access control unit and a peripheral device.

Within an **Exchange**, for example, it is possible to calculate the time it took the peripheral device to response to the command. This is useful in profiling the performance of a reader. This is just one example of the analysis possibilities of the **osdpspy** analysis tool.

As the **Exchanges** are assembled, they can be offered up for consumption by the **Exchange Logger** component.

5.4.2.4 Exchange Logger

As previously stated, the **Exchange** is the basic unit of analysis for the prototype. The purpose of the **Exchange Logger** component is to analyse each **Exchange** and create an **Annotation**. An annotation contains the following information:

- A string containing a structured log message containing a set of key items.
- An array of objects corresponding to the key items in the log message.

This data is in a form readily passed to the Serilog structured event log library.

The **Annotation** starts with the basic information in the **Exchange**. The **Exchange Logger** component then passes the **Exchange** and **Annotation** through a serious of **Annotators** to successively enrich the **Annotation**. The role of the

Annotator will be explained more fully in the next section. Once the **Exchange** has been fully processed, the resulting **Annotation** is sent to the structured event logger of choice: Seq, Elasticsearch or Splunk.

5.4.2.5 Annotators

Whilst an **Exchange** is the unit of analysis for the prototype, an **Annotator** component is a fragment of the software that performs a specific, focussed analysis of an **Exchange** from which it enriches the supplied **Annotation**. By passing an **Exchange** through a series of **Annotators** it is possible to build a rich set of information about protocol activity that can be extensively queried in the chosen structured logging tool.

As we shall see in the examples presented below, an **Annotator** need not just concerned with a single **Exchange**, it may also maintain state across many **Exchanges**. We will use this particular facet to capture files transferred using the **osdp_FILETRANSFER** command and decrypt the secure channel session.

When new protocol analysis objectives are formulated, the idea is that an **Annotator** is implemented to fulfil those objectives.

We now present four of the **Annotators** implemented in the **osdpspy** analysis prototype:

- Raw Frame Annotator
- Secure Channel Annotator
- Valid Access Annotator
- File Transfer Annotator

5.4.2.6 Raw Frame Annotator

The **Raw Frame Annotator** is the first **Annotator** executed by the prototype and processes a single **Exchange**. It breaks out the basic information about the exchange in human readable form including:

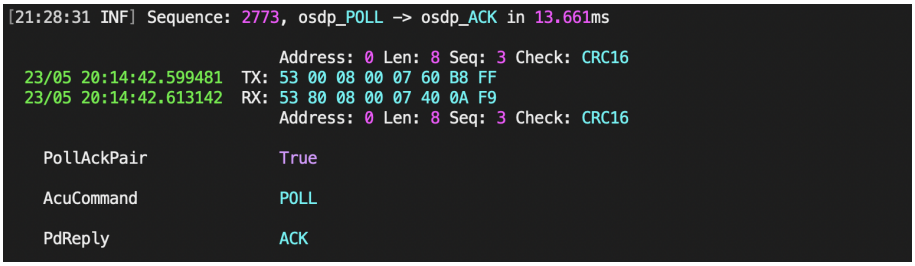

Figure 5.2: *osdp_POLL/osdp_ACK Pair in **osdpspy** Output*

osdpspy presents trace data in the form shown in Figure 5.2. Here we see the basic **osdp_POLL**/**osdp_ACK** pair that is the most common command and reply seen in a running protocol. Since this exchange is so common, we tag it with a **PollAckPair** attribute set to **True** so that the data tools that we can use (i.e. Seq or Elasticsearch) can easily filter out these exchanges. The **-f** or **–filter** options can also be used to filter out these exchanges at source.

In this simple example, we already see some of the annotation work that **osdpspy** performs, including the following:

- A **Sequence** number is allocated as a unique identifier for each exchange, in this case **2773**.
- The command and reply codes are decoded and presented as the corresponding names.
- The time to complete the exchange is calculated and presented, in this case 13.661 mS.
- The frame header values are extracted and presented for both the command and the reply.
- Any special tags (e.g. **PollAckPair**) for this exchange are presented next.
- Grouped with **AcuCommand** tag we will find the payload for the command and, where possible, a number of tags containing the decoded data for the command. In the case of an **osdp_POLL** command, there is no payload to display or decode. These attributes are generated by the Command Annotator.
- Finally, grouped with the **PdReply** tag, we will find the payload for the reply and, where possible, a number of tags containing the decoded data

for the reply. In the case of an **osdp_ACK** reply, there is no payload to display or decode. The attributes are generated by the Reply Annotator.

5.4.2.7 Secure Channel Annotator

Figure 5.3: Secure Channel Setup Using Default SCBK

The central problem of capturing the key exchange and decrypting the protocol is largely solved in the 171 lines of C# that implement the **Secure Channel Annotator** component.

The responsibility of this component is to listen for commands that affect the state of the secure channel, maintain the RMAC from frame to frame, decrypt the contents of the frame and convert this data to useful information. The annotator also reports the keys that are used to establish the session.

This is a good example of the **Annotator** concept which is used as the main analysis tool in the **osdpspy** prototype, exhibiting the following properties:

- Maintains state across multiple **Exchanges**.
- Focussed on a single objective: decrypting secure channel in this case.
- Compact implementation.

In Figure 5.3, we see the annotated output from **osdpspy** showing the establishment of the secure channel followed by the **osdp_KEYSET** command to set the Secure Channel Base Key (SCBK). The Command and Reply Annotators expand the command and reply information and, as we can see in the **osdp_KEYSET**/**osdp_ACK** exchange, are decrypting the secure channel payloads and revealing the test key of **11111111111111111111111111111111** used by the test software we were monitoring.

As keys are recovered they are save in a ket store and persisted to disk. This allows **osdpspy** to use the secure channel base key saved against a specific reader when there is no default base key activity in the protocol.

5.4.2.8 Valid Access Annotator

In Section 4.6 we outlined a number of objectives for testing a real-world access control system. The **Valid Access Annotator** seeks to address the following:

1. Capture and report encrypted card data.
2. Capture and report encrypted PIN entry.
3. Capture card numbers, complete PINs and report triggered by an **osdp_LED** command.

```
[23:20:08 INF] Sequence: 54824, osdp_POLL -> osdp_RAW in 49.075ms

                        Address: 0 Len: 14 Seq: 2 Check: CRC16 Security: 02 15
 13/07 22:20:08.128310  TX: 53 00 0E 00 0E 02 15 60 30 C2 A7 D9 79 7B
 13/07 22:20:08.177385  RX: 53 80 1E 00 0E 02 18 50 2E 4E 64 4F 0B 6C 09 52 75 0C 01 C4 3B 46 0D 5F F9 D7 CB 3C 7D 1E
                        Address: 0 Len: 30 Seq: 2 Check: CRC16 Security: 02 18


  AcuCommand            POLL

  PdReply               RAW
  PdCipher              2E 4E 64 4F 0B 6C 09 52 75 0C 01 C4 3B 46 0D 5F
  PdPlain               00 01 38 00 00 48 04 02 00 09 85 80 00 00 00 00
  ReaderNumber          0
  FormatCode            P/DATA/P, Wiegand
  BitCount              56
  Data                  00 48 04 02 00 09 85
```

Figure 5.4: Encrypted Card Data, Decrypted by **osdpspy**

In Figure 5.4 we an **osdp_RAW** response to the **osdp_POLL** command containing the data read from the card. **osdpspy** has decrypted and decoded the data as a 56-bit Wiegand format containing a bit-stream of **00 48 04 02 00 09 85**. For the sake of simplicity, I have used a BCD card format and in Figure 5.5, we see the number printed on the actual card.

Figure 5.5: Actual Test Card Used

We see that the card number reported by **osdpspy**, with the addition of a leading zero, is the same as the number printed on the card.

As shown in Figure 5.6, capturing a PIN number, in this case **1234**, is straightforward once **osdpspy** is decrypting the communication between the access control unit and the reader.

Finally, to fulfil the objectives of the **Valid Access Annotator**, we need a trigger. It is a widely used convention that the reader LED should turn green when access is granted at a door. To achieve this, we inspect the **osdp_LED** commands as they are issued by the access control unit. In Figure 5.7, we see an example of an **osdp_LED** command indicating access at a door.

We consider access to have been granted if the following conditions are met:

- The command contains a **TemporaryControlCode** value of **Set**.
- Either of the **TemporaryOnColor** or **TemporaryOffColor** values are set to **Green**.
- Either of the **PermanentOnColor** or **PermanentOffColor** values are set to **Green**.

Now that we can capture card numbers, PIN numbers and identify when access has been granted, we can start to assemble the **Valid Access Annotator**. This annotator builds a model of what is happening at the reader by:

Figure 5.6: Encrypted PIN Data, Decrypted by **osdpspy**

- Capturing card reads reported by the reader in the **osdp_RAW** replies.
- Capturing PIN data reported by the reader in the **osdp_KEYPAD** replies.
- Time-stamping Card and PIN data and storing this data in a list as it is detected.
- Detecting a Green LED in an **osdp_LED** as the trigger to process the card and PIN number data.
- Reporting the last card data and any key presses in the last ten seconds as an **Alert**.

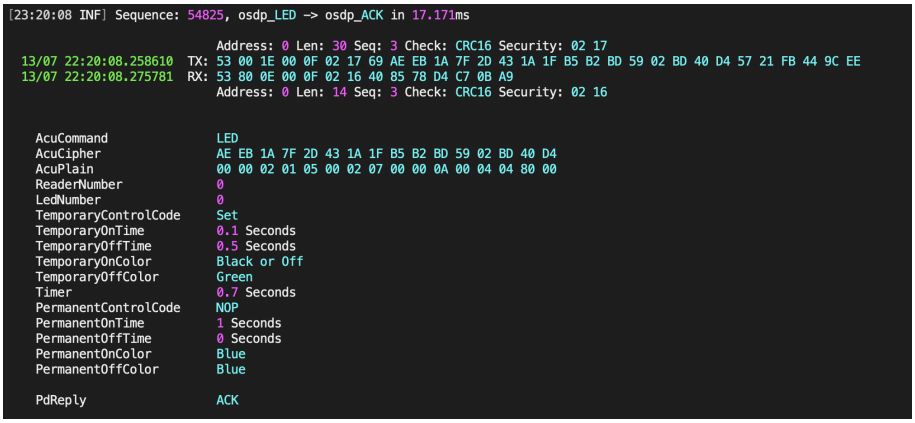

Figure 5.7: Green LED Command

In Figure 5.8, we see how this annotator reports its findings. We see that **osdp-spy** has decrypted the OSDP secure channel, extracted card and PIN data and converted this data into **actionable intelligence**: the PIN number required by a card to gain access at a door.

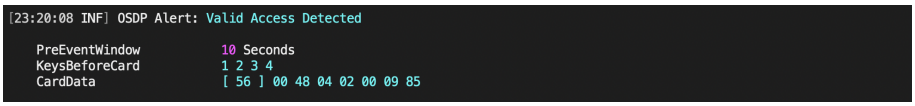

Figure 5.8: OSDP Alert: Valid Access Detected

5.4.2.9 File Transfer Annotator

Whilst OSDP file transfer has been in use for around five years, this feature was only formalised with the introduction of IEC 60839-11-5:2020 in 2020. File transfer allows an access control unit to transfer a binary image to a reader without any knowledge of the contents. The reader must first determine if it can process the file because these files are device-specific.

Some examples of the information contained in an OSDP file for a reader to process are:

- Configuration data
- Encryption keys
- Device firmware

In Figure 5.9, we see that **osdpspy** can decrypt individual **osdp_FILETRANSFER** messages.

Figure 5.9: Encrypted **osdp_FILETRANSFER** Message

OSDP file transfer works by segmenting the file into blocks of data to be passed in sequence to a reader using the **osdp_FILETRANSFER** command. The **File Transfer Annotator** extracts these blocks, reassembles the file and, at the request of the user, saves the file to disk for further analysis.

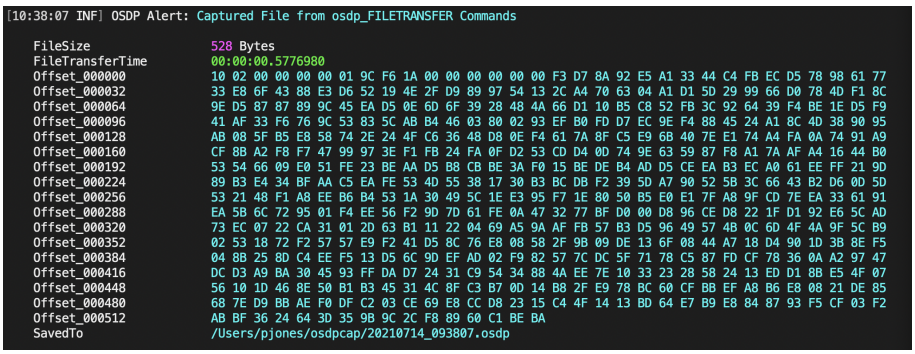

Figure 5.10: OSDP Alert: Captured File from **osdp_FILETRANSFER** Commands

In Figure 5.10, we see how the **File Transfer Annotator** generates an alert to show the user that a file has been captured.

We could infer the following from this trace:

- The first 16 bytes appear to be in plain text.
- At offset 6 we find the OUI, an IEEE vendor identifier, of the reader used in the transfer.

- The size of the file is 528 bytes. At offset 0 we we find **10 02**. This is the value 528 in little endian.
- The overall size of the file is a multiple of 16 bytes. We a have a 16 byte header. We can reasonably infer that the remainder of the file is encrypted.

Even though further work is required to get value from the file that has been captured, **osdpspy** has once again turned data captured from an OSDP bus into **actionable intelligence**.

5.4.2.10 Further Annotators

Further examples of **Annotators** can be found in the dissertation source code.

5.4.3 Outputs

Once we have manufactured a rich seam of information from a sequence of OSDP protocol frames, we have a choice of the following outputs:

- To the console.
- To an **osdpcap** file.
- Files captured by monitoring the **osdp_FILETRANSFER** command.
- To a Seq event sink.
- To an Elasticsearch event sink.

The various outputs from the application are selected and specified by command line options.

5.4.3.1 Console

The console always receives output from the application. When the first implementation was used for the first time, it was noticed that the protocol spends

most of its time sending an **osdp_POLL** command and receiving an **osdp_ACK** response. This made it difficult to spot activity such as card reads or key presses on the bus.

To remedy this problem, an option to filter out **osdp_POLL**/**osdp_ACK** pairs was added to the command line:

osdpspy listen -f *[other options]*

We have already seen numerous examples of console output from **osdpspy** in this chapter.

5.4.3.2 osdpcap File

For interoperability with other tools that use the **osdpcap** file format to process sequences of OSDP frames, we have added the option to save a trace to an **osdpcap** file when the tool is listening to a live OSDP bus:

osdpspy listen -c[:<capture-directory>] *[other options]*

Where **<capture-directory>** is the path to the directory where the **osdpcap** files should be created. If this path is omitted, the files are created in the current working directory. A log file with a name of the form **YYYY-MM-DD-hhmmss.osdpcap** is created to which the received OSDP frames are logged.

5.4.3.3 File Transfer Capture

The application has the option to save files to disk that have been transferred from the access control unit to the peripheral device by segmenting the file and sending each fragment using the **osdp_FILETRANSFER** command. This option works with both types of input: **osdpcap** file or live OSDP bus.

To extract files from either of these source, we use the following command:

osdpspy listen -t[:<capture-directory>] *[other options]*

Where **<capture-directory>** is the path to the directory where the OSDP file trasnfer files should be created. If this path is omitted, the files are created in the current working directory. A file with a name of the form **YYYY-MM-DD-**

hhmmss.osdp is created to which the received OSDP file is saved.

5.4.3.4 Seq

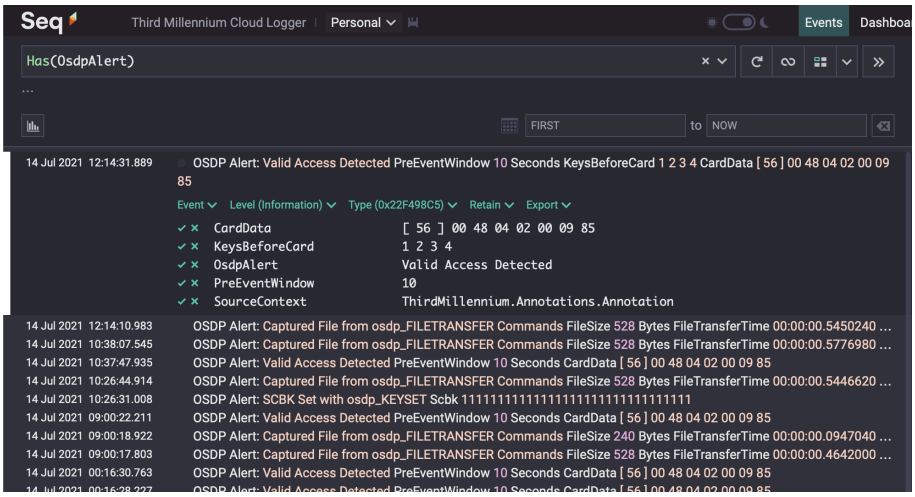

Figure 5.11: Example Seq Output

Third Millennium often uses Seq for event logging to the cloud from various applications that it has developed. Seq is easy to install and use but only has basic security. Security of the log file is not generally a concern when the company is developing prototypes and, as a consequence Seq was the first event sink that was implemented with **osdpspy**.

An industry standard event sink called Elasticsearch has been implemented to demonstrate compatibility with commercial SIEM solutions as a future pathway for the development of **osdpspy**. Another commercial tool call Splunk will be implemented in a future version.

By logging the information we derive from **osdpspy** to an event sink, we have the opportunity to use the rich query and dashboard capabilities that these products provide.

The command line for logging to Seq is as follows:

osdpspy listen -s <seq-url> *[other options]*

Where **<seq-url>** is the URL of the Seq instance you wish to log to.

Figure 5.11 shows an example of the output displayed by Seq. In this case, the data has been filter to only show the alerts generated by **osdpspy**.

5.4.3.5 Elasticsearch

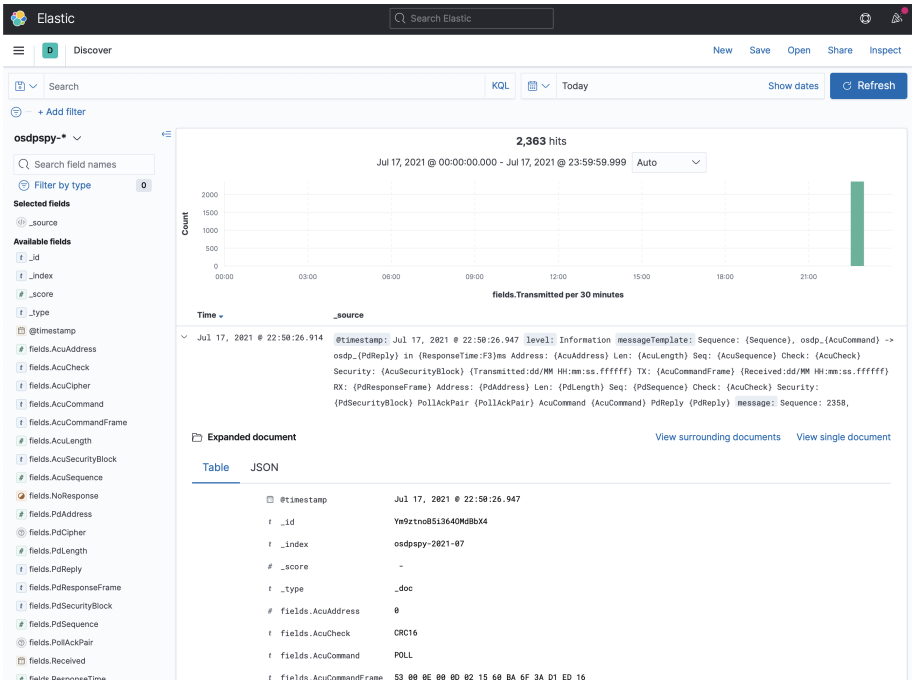

Figure 5.12: Elasticsearch Data Ingestion from **osdpspy**

The command line for logging to Elasticsearch is as follows:

osdpspy listen -e <elasticsearch-url> *[other options]*

5.5 Other Functions

For the convenience of the user, additional commands and options have been provided with the **osdpspy** analysis tool.

5.5.1 List Subcommand

The **list** subcommand is provide to allow the user to identify which USB serial ports are available to the application and their device names. The user can then select which device is used with the **listen** subcommand.

To obtain a list of the available USB devices attached to the computer, use the following command:

osdpspy list

5.6 Command Line Summary

In Table 5.1 we summarise all of the commands and options available to the **osdpspy** prototype analysis tool. We note that each of the options has both a short name (i.e. **-e**) and a long name (i.e. **–elasticsearch**) that is more descriptive. Either may be used when entering a command line.

Command	Sub-Cmd.	Opt.	Long Name	Description
osdpspy	import	-e	–elasticsearch	Log to Elasticsearch
		-f	–filter	Filter POLL/ACK
		-i	–input	osdpcap file
		-s	–seq	Log to Seq
		-t	–filetransfer	Capture transferred files
	list			List Serial Devices
	listen	-c	–capture	Trace to **osdpcap**
		-e	–elasticsearch	Log to Elasticsearch
		-f	–filter	Filter POL/ACK
		-p	–port	Serial Device
		-r	–rate	Baud Rate
		-s	–seq	Log to Seq
		-t	–filetransfer	Capture transferred files

Table 5.1: *osdpspy Command Line Summary*

5.7 Deliverables

The source code for **osdpspy** is provided in a separate file called OsdpSpy.ZIP and contains a standalone Visual Studio 2019 project that allows us to build the application. The main application project can be found in the **ThirdMillennium.OsdpSpy** folder.

Before we can install the application on your machine, we first need to install .NET 5 or later from **dotnet.microsoft.com/download** (Microsoft Corporation, 2021). Once the appropriate version of .NET has been installed, we can install **osdpspy** using the following command:

dotnet tool install -g thirdmillennium.osdpspy –add-source ./nupkg

Once the installation is complete, we can run the following command to verify the installation:

osdpspy -v

osdpspy will report a version of **0.0.4** if it has been installed correctly.

5.8 Conclusion

In this chapter, we have shown how we have constructed a software tool called **osdpspy**. We have show the structure and operation of the application. We have also shown how to use **osdpspy** to capture data leaked from the secure channel to meet the objectives of the test plan presented in the previous chapter.

Chapter 6

Analysis

6.1 Introduction

In this chapter, we evaluate the OSDP secure channel implementations of two commercially available access control systems. We detail the framework that we are going to apply to these systems, perform our analysis and then detail our findings for each of the systems.

The **osdpspy** protocol analysis tool that we have developed in the previous chapter allows us to monitor and, under certain circumstances, decrypt the secure channel traffic and extract useful information and actionable intelligence. We have chosen to apply **osdpspy** to two different systems in order to demonstrate that it is a general purpose tool for assessing all OSDP-based access control systems.

6.2 Analysis Framework

In this section we provide the background to each of the steps we take to evaluate each of the systems against the outline objectives we set in Section 4.6. This framework provides a foundation on which a more extensive penetration testing frame work for OSDP secure channel implementation could be developed.

6.2.1 System Hardware

In this section we provide a brief description of the system, focussing on the access control unit and the reader. We also provide a picture of the units under test.

6.2.2 Setting Up Secure Channel

Pairing is the process by which the secure channel is setup between the access control unit and the reader or other peripheral device. The reader must be enabled for pairing so that it accepts the default secure channel base key (SCBK). The controller is then set by some means to start secure channel with the default SCBK.

Each access control unit and reader vendor have their own specific methods to enable pairing but once initiated, most system execute the following process:

1. Start secure channel with the default SCBK.
2. Set the actual SCBK in the reader or peripheral device.
3. Restart secure channel with the newly set SCBK.

For each of these cases we will use a 3millID Blue Diamond reader, a Third Millennium design.

6.2.3 Detecting Key Exchange

If we are successful inititiating the key exchange we should be able to capture and evidence this with a trace from the **osdpspy** analysis tool. Our trace should show the steps outlined above.

6.2.4 Capture And Report Encrypted Card Data

Once the secure channel has been established, we will be successfully decrypting the secure channel session in real-time. We can then capture the system

behaviour when a valid card is presented to the reader. If successful, we evidence the decrypted card data in the **osdp_RAW** response from the controller in an **osdpspy** trace. We can also confirm that the card number is the card number that was expected.

6.2.5 Capture and Report PIN Entry

We can also capture the system behaviour when a key is pressed on the reader. As with the card data, we can evidence the decrypted keypad data in the **osdp_KEYPAD** response from the controller in an **osdpspy** trace.

6.2.6 Capture A Valid Access

Once we have successfully captured card and PIN data, we can attempt to use **osdpspy** to capture a valid access to the system where a card and a PIN number have been used. In Section 5.4.2.8, we described the **Valid Access Annotator** which assembles the card and PIN data and raises an alert when a green LED is presented at the reader. We evidence that we have successfully captured a card and PIN number with a **Valid Access Detected** alert in an **osdpspy** trace.

6.2.7 Capture File Transfer

OSDP file transfer is a relatively new feature and is not yet widely supported. If the system under test supports the feature, we transfer a reader configuration file from the access control unit to the reader in a secure channel session.

Each system that supports OSDP file transfer will have its own method for starting the process. Once started, osdpspy will decrypt and assemble the file being transferred and evidence success with a **Captured File from osdp_FILETRANSFER Commands** alert in the trace.

Finally, with the file saved in the capture directory, we can verify the captured file against the original configuration file to confirm that the analysis objective has been met.

6.2.8 Summary of Findings

Once we have completed the analysis, we summarise the results of the evaluation of the system in this section.

This is also an opportunity to provide additional observations of the the protocol implementation of both the access control unit and the reader used in the evaluation.

6.3 Nedap AEOS

Nedap is a Dutch multinational technology company based in Groenlo, Netherlands. They produce an access control system called AEOS and the access control unit we will be testing is the AP7803m controller.

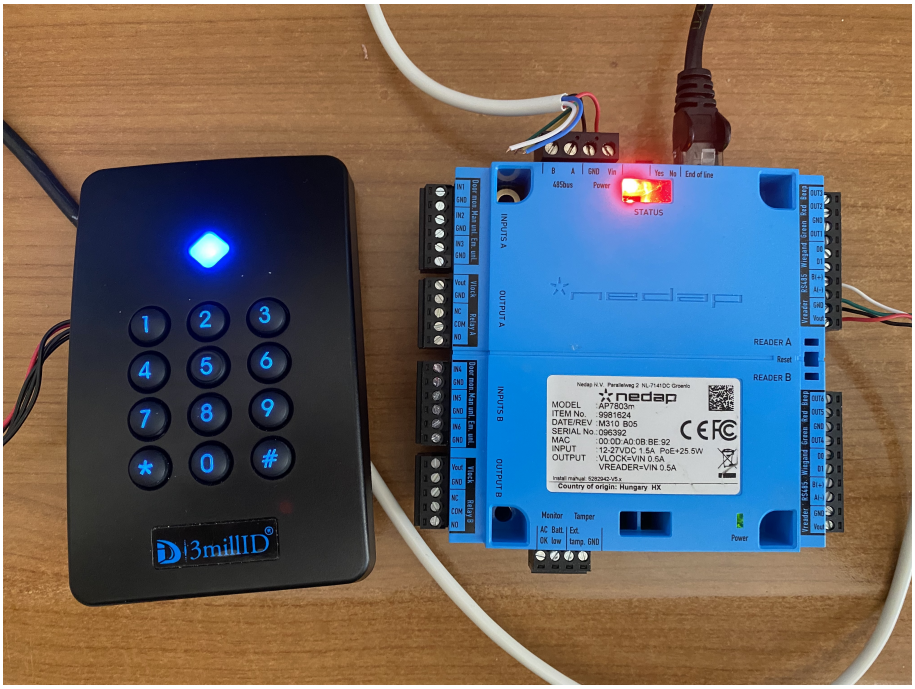

Figure 6.1: *Nedap AP7803m Controller and 3millID Blue Diamond Reader*

In Figure 6.1 we see the access control unit and reader that we are going to observe with the **osdpspy** OSDP analysis tool.

For the purposes of this evaluation, we will be using a card with a card number of **0160402081398** and a PIN number of **1 2 3 4**.

6.3.1 Setting Up Secure Channel

The process for setting up secure channel with this access control unit is as follows:

1. Configure the access control unit for secure channel.
2. Enable the reader for pairing.
3. Pair the Reader and Access Control Unit

6.3.1.1 Configure The Access Control Unit for Secure Channel

In Figure 6.2 we see the **AEmon** controller management software and the configuration dialog for setting the secure channel base key from the AEOS controller. The dialog asks for an "encryption key" and we have entered a test key of **00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF**. One would expect this to be the secure channel base key used by the controller but, as we shall see, this is not the case.

To complete the configuration, we deploy the new configuration to the access control unit from AEmon utility using the **Configuration/Deploy...** command from the menu.

6.3.1.2 Enable Reader for Pairing

The Blue Diamond reader comes with special configuration cards that are used to configure the OSDP settings of the reader. On of these cards, the **BD_SECURE** card, resets the secure channel base key stored in the reader and enables the reader for pairing: the controller may used the default secure channel base key (SCBK) to load the new SCBK.

To enable the reader for pairing, present the configuration cards in the following order:

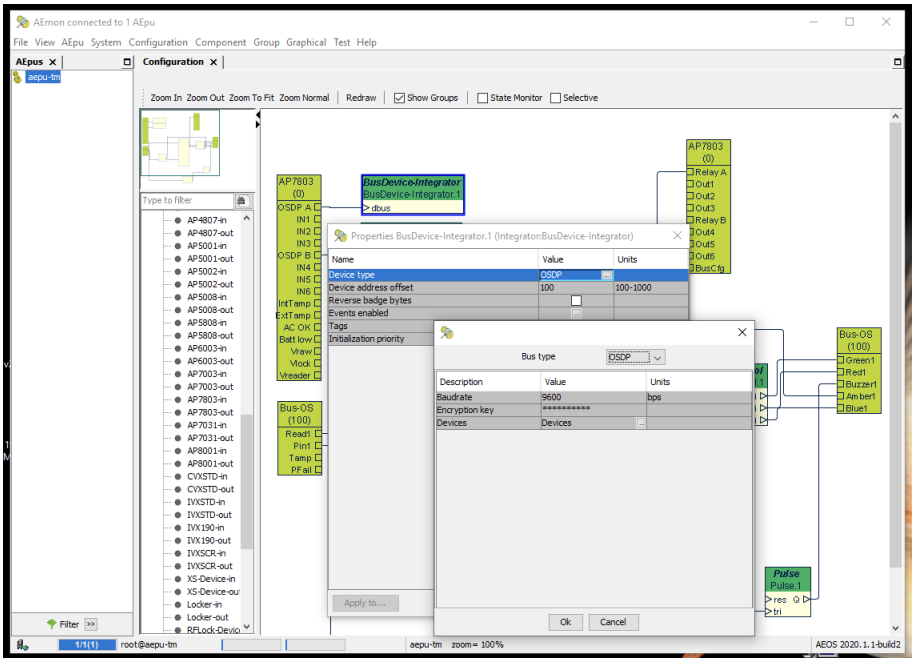

Figure 6.2: *Configuring the Nedap AP7803m Controller*

- Present the **BD_UNLOCK** card to enable the reader to accept configuration cards.
- Present the **BD_SECURE** card to clear an existing SCBK and enable pairing.

6.3.1.3 Pair The Reader And Access Control Unit

After setting up the access control unit for secure channel and the reader for pairing with the controller, we need to initiate the pairing process. After some experimentation, we found that if you deploy the configuration to the controller, this is not enough to initiate pairing. We found that a power cycle of the controller was required to start the process.

6.3.2 Detecting Key Exchange

After trying to pair with the expected secure channel base key, the access control units sets up a secure channel session with the default SCBK and sets the

private SCBK in the reader using the **osdp_KEYSET** command as shown in Figure 6.3.

Figure 6.3: *AEOS Key Setup with the Default SCBK*

We see that the access control unit sets an actual SCBK of **49 F2 EF 84 2B A7 A5 01 93 B0 17 8A 12 A6 28 EC** and differs from the key that was entered into the configuration software.

The controller then sets the secure channel base key and then calls **osdp_POLL** for the first time since the power was cycled as shown in Figure 6.4.

In the decrypted **osdp_LSTATR** response to the **osdp_POLL** we see that the reader has just been reset because of the power cycle to the reader and the controller.

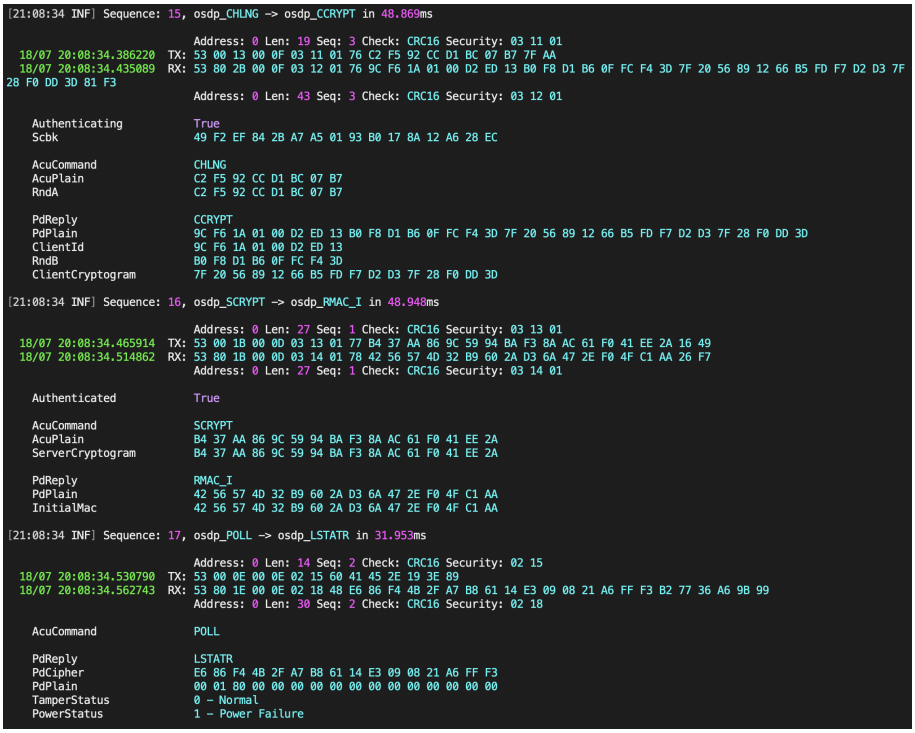

Figure 6.4: *AEOS Establish the Secure Session with the Shared SCBK*

6.3.3 Capture And Report Encrypted Card Data

The first thing we need to do in working towards our goal of detecting a valid access is to verify that we are reading and decrypting the card number from the card correctly. In Figure 6.5 we see that the card number has been successfully decrypted and extracted from the **osdp_RAW** response from the access control unit by **osdpspy**.

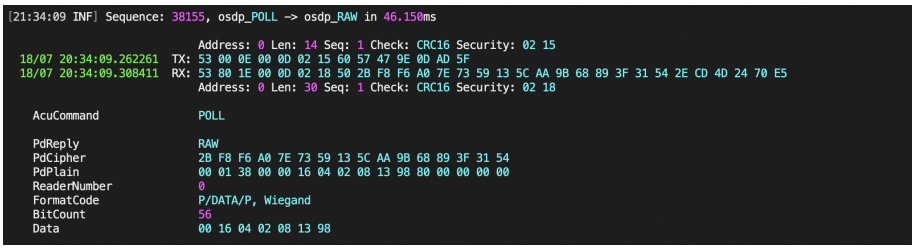

Figure 6.5: *Card Data Read Monitoring AEOS*

6.3.4 Capture and Report PIN Entry

The next step is to verify that we are decrypting the PIN data correctly. In Figure 6.6 we see that ta key press, a number **4**, has been successfully decrypted and extracted from the **osdp_KEYPAD** response from the access control unit.

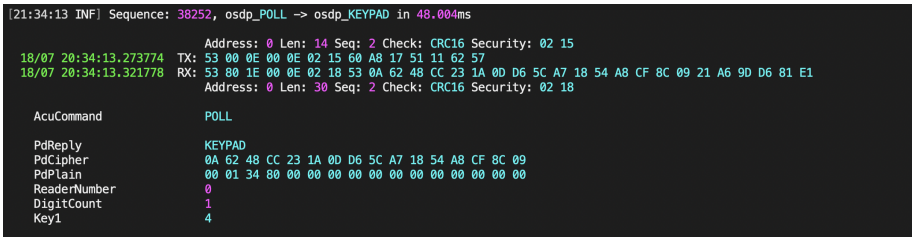

Figure 6.6: *PIN Entry Read Monitoring AEOS*

6.3.5 Capture A Valid Access

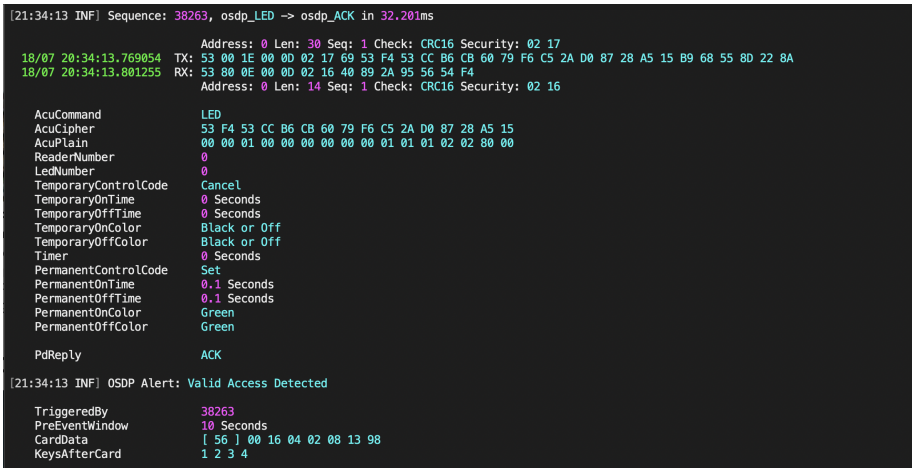

Figure 6.7: *Valid Access Detected Monitoring AEOS*

In Section 5.4.2.8 we described the function of the **Valid Access Annotator**. This annotator collects card and PIN data to be processed when a green LED is requested for the reader. In our initial implementation, we had assumed that only the temporary LED values of the **osdp_LED** command would be used for this detection. In working with the AEOS controller, we found that this was not the case and that AEOS uses the permanent LED values. This finding influenced the implememtation of the annotator to allow us to complete this analysis.

In Figure 6.7, we see the **osdp_LED** command that triggered processing by the **Valid Access Annotator**. We confirm that the permanent LED colours of the command have been used to set the reader LED to green and that **osdpspy** has correctly detected that we have used a card number of **0160402081398** and a PIN number of **1 2 3 4** to gain access at the door.

6.3.6 Capture File Transfer

The version of AEOS used for the tests presented in this section does not support OSDP file transfer. Nedap were contacted during the evaluation and indicated that OSDP file transfer is on their roadmap for a future release.

6.3.7 Summary of Findings

The analyses presented in this chapter are as much a test of **osdpspy** as the unit under test. Through our analysis of the AEOS access control unit we have found that:

- The LED may be set to green using the permanent parameters of the **osdp_LED** command.
- **osdpspy** performed as expected with the exception of the LED problem.
- The key entered in the user interface of AEOS is not the key that is used as the SCBK at the reader.
- We successfully decrypted the card data.
- We successfully decrypted PIN data.
- We successfully detected a valid access and revealed the card number and the corresponding PIN number.

The choice by the implementers of AEOS to use a different key to the one entered in the user interface for the SCBK is a good choice. This provides some protection against the actual SCBK being shared with bad actors.

6.3.7.1 Additional Observations

In addition to meeting the analysis objectives, with the exception of OSDP file transfer, we also discovered incorrect behaviour in the implementation of the protocol by both the controller and the reader.

The AEOS controller has an incorrect approach to retries and fails to roll the sequence number before issuing a new command. In secure channel, this causes the secure channel session to break and the session restarted. This issue has been referred to Nedap.

The issue with the controller also highlighted a problem with the Blue Diamond reader where it was not detecting a new command on the same sequence. This issue has now been fixed in the Blue Diamond reader, a Third Millennium design.

6.4 Lenel S2 On Guard

Lenel S2 is subsidiary of the Carrier group and one of the leading manufacturers of access control systems in the world. For this evaluation, we will be using their On Guard 8 access control platform with a LNL-X2210 controller.

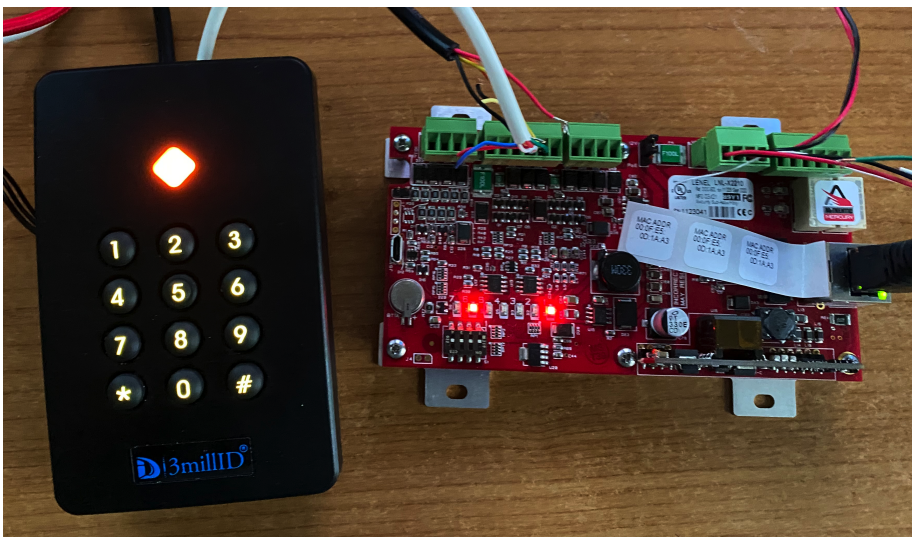

Figure 6.8: *LenelS2 X2210 Controller and 3millID Blue Diamond Reader*

In Figure 6.8 we see the access control unit and reader that will be evaluated in this section.

6.4.1 Setting Up Secure Channel

As before, the process for setting up secure channel with this access control unit is as follows:

1. Configure the access control unit for secure channel.
2. Enable the reader for pairing.
3. Pair the Reader and Access Control Unit

6.4.1.1 Configure The Access Control Unit for Secure Channel

The first step to enabling secure channel on the Lenel controller is to configure the reader for secure channel. This is achieved by running the System Administration application and checking the **Secure Channel** check box in the reader dialog as shown in Figure 6.9.

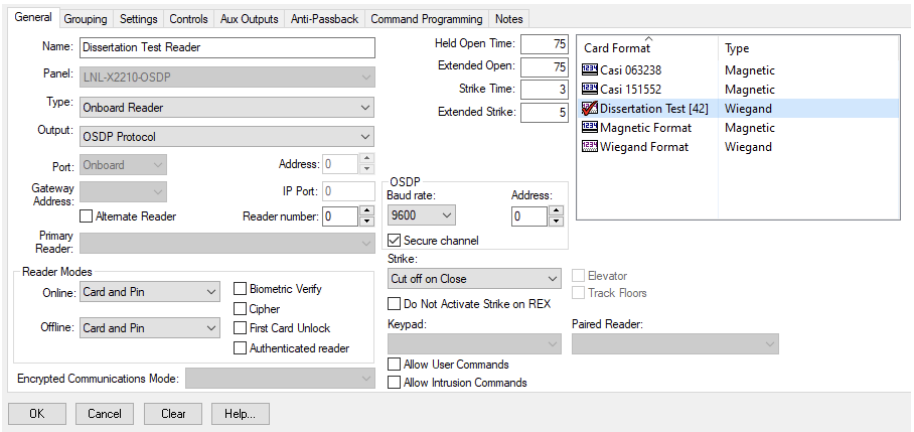

Figure 6.9: *Configuring LenelS2 X2210 Controller*

6.4.1.2 Enable Reader for Pairing

The Blue Diamond reader comes with special configuration cards that are used to configure the OSDP settings of the reader. On of these cards, the **BD_SECURE** card, resets the secure channel base key stored in the reader and enables the reader for pairing: the controller may used the default secure channel base key (SCBK) to load the new SCBK.

To enable the reader for pairing, present the configuration cards in the following order:

- Present the **BD_UNLOCK** card to enable the reader to accept configuration cards.
- Present the **BD_SECURE** card to clear an existing SCBK and enable pairing.

6.4.1.3 Pair The Reader And Access Control Unit

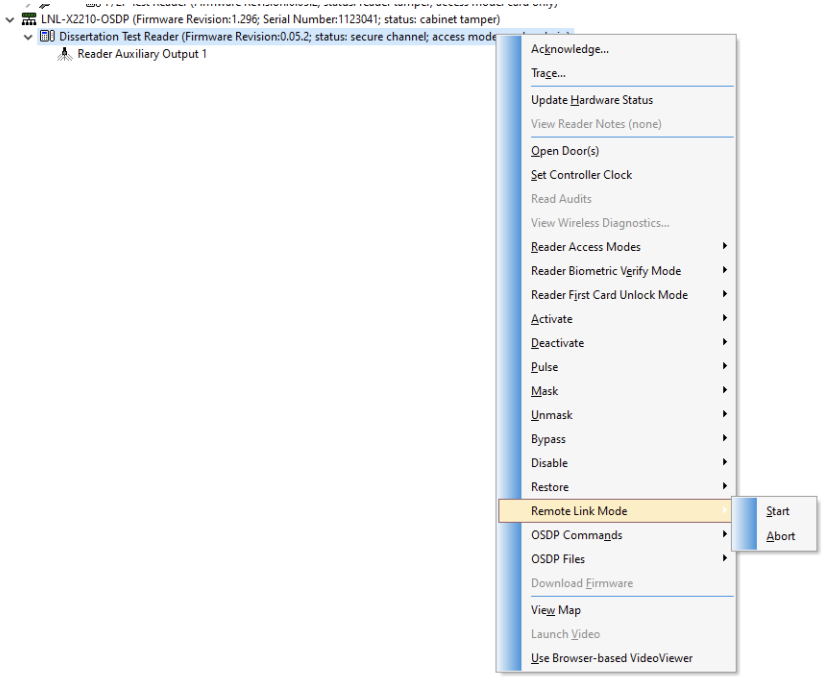

Figure 6.10: *Configuring LenelS2 X2210 Controller*

After setting up the access control unit for secure channel and the reader for pairing with the controller, we need to initiate the pairing process. In the Lenel On Guard system, this is achieved by running the Alarm Monitoring Application and starting "Remote Link Mode" for the reader.

In Figure 6.10 we the see the menu to start what the On Guard application calls the Remote Link Mode. When we click on **Start**, a command is sent to the access control unit to start the secure channel to the reader.

6.4.2 Detecting Key Exchange

After trying to pair with the expected secure channel base key, the access control units sets up a secure channel session with the default SCBK and sets the private SCBK in the reader using the **osdp_KEYSET** command as shown in Figure 6.11.

Figure 6.11: *OnGuard Key Setup with the Default SCBK*

We see that the access control unit sets an actual SCBK of **D9 CD AA 7A F9 C2 EA 82 19 F3 C1 4E 26 88 34 16**. At no time have we entered a key value and we believe this key value is generated by the system based on the unique identifier provided by the reader. Given the widespread use of this controller in the market, this is an interesting topic for further research. Is it possible to determine the secure channel base key that will be used by one of these controllers for a given reader identifier?

6.4.3 Capture And Report Encrypted Card Data

In Figure 6.12 we see that decrypted card data has been captured from the secure channel.

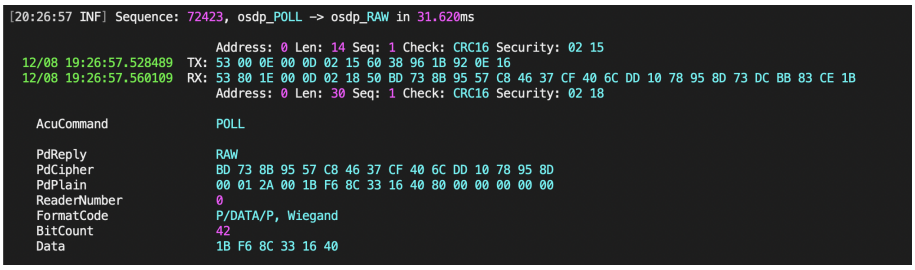

Figure 6.12: *Card Data Read Monitoring On Guard*

Unlike the previous evaluation, Lenel On Guard does not support decoding of BCD data. To overcome this we used a custom Wiegand format to pack the binary card number into a 42-bit Wiegand frame. This process is performed by the reader under test. We discussed Wiegand formats in Section 3.2.1.4.

Given that the number is being sent over OSDP with its inherent integrity checks, the card number is sent as a simple 42-bit binary number with no parity bits.

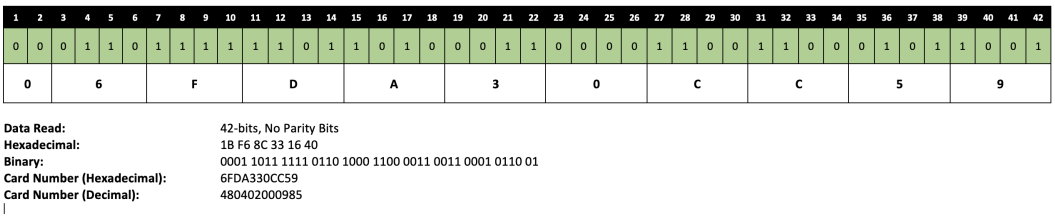

Figure 6.13: *Decoding the 42-Bit Card Data*

In Figure 6.13 we see how the data is further decoded to reveal the actual card

number, in this case **480402000985**. If we prefix this number with a leading zero, we end up with the same number printed on the card.

Third Millennium has an extensive library of Wiegand formats gathered over the last 30 years. An annotator can be written to automatically decode the data into matching formats for further analysis. Given the commercial sensitivity of this information, it has not been included in **osdpspy** at this time.

6.4.4 Capture and Report PIN Entry

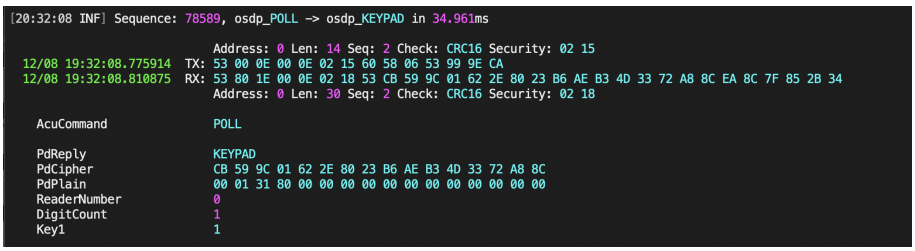

Figure 6.14: *PIN Data Read Monitoring On Guard*

In Figure 6.14 we see that decrypted PIN data has been successfully captured from the secure channel by **osdpspy**.

6.4.5 Capture A Valid Access

In Figure 6.15 we see that a valid access to the system has been successfully captured from the secure channel by **osdpspy**.

6.4.6 Capture File Transfer

The Lenel On Guard access control system supports OSDP file transfer. Files are specific to a particular manufacturer of access control reader and are stored in the access control unit. To transfer a file to a reader, the user must initiate the file transfer to the reader by selecting the file they want to send.

In Figure 6.16 we show that we can send the configuration file called **Reader Configuration.AVC** to our **Dissertation Test Reader** by clicking on the **Transfer** menu item.

```
[20:34:57 INF] Sequence: 81925, osdp_LED -> osdp_ACK in 15.922ms

                          Address: 0 Len: 30 Seq: 2 Check: CRC16 Security: 02 17
  12/08 19:34:56.986691  TX: 53 00 1E 00 0E 02 17 69 DC 50 7C 20 D0 83 8C AB 90 C9 FA 02 EA C3 A2 C1 37 AB 6C 7A 32 D3
  12/08 19:34:57.002613  RX: 53 80 0E 00 0E 02 16 40 82 63 F4 97 74 50
                          Address: 0 Len: 14 Seq: 2 Check: CRC16 Security: 02 16

    AcuCommand              LED
    AcuCipher               DC 50 7C 20 D0 83 8C AB 90 C9 FA 02 EA C3 A2 C1
    AcuPlain                00 00 02 03 00 02 02 1E 00 01 32 32 04 04 80 00
    ReaderNumber            0
    LedNumber               0
    TemporaryControlCode    Set
    TemporaryOnTime         0.3 Seconds
    TemporaryOffTime        0 Seconds
    TemporaryOnColor        Green
    TemporaryOffColor       Green
    Timer                   3 Seconds
    PermanentControlCode    Set
    PermanentOnTime         5 Seconds
    PermanentOffTime        5 Seconds
    PermanentOnColor        Blue
    PermanentOffColor       Blue

    PdReply                 ACK

[20:34:57 INF] OSDP Alert: Valid Access Detected

    TriggeredBy             81925
    PreEventWindow          10 Seconds
    CardData                [ 42 ] 1B F6 8C 33 16 40
    KeysAfterCard           1 2 3 4
```

Figure 6.15: *Valid Access Detected Monitoring On Guard*

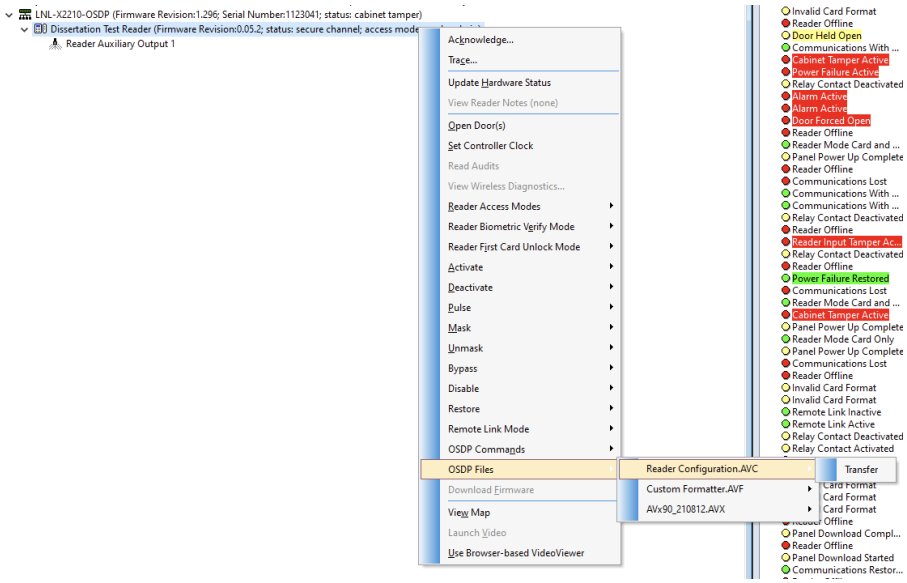

Figure 6.16: *Starting a File Transfer in On Guard*

Once this item has been been clicked, we see OSDP file transfer activity on the OSDP bus.

In Figure 6.17 we show the last fragment of the file transfer being passed from the access control unit to the reader to evidence that the transfer took place in the secure channel.

Figure 6.17: *Encrypted File Transfer Monitoring On Guard*

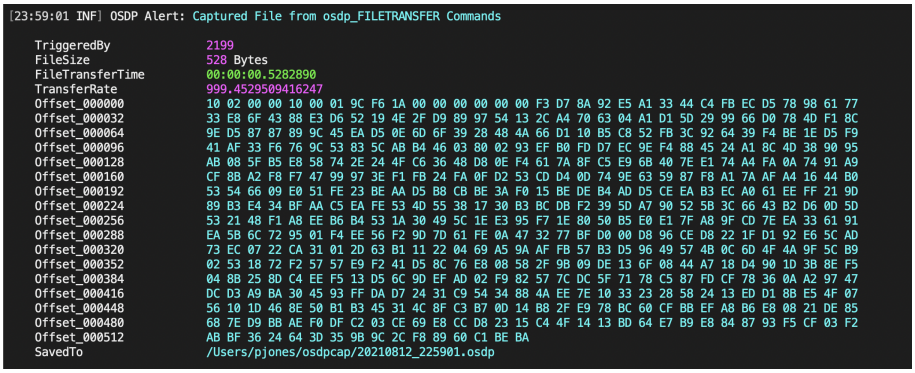

Figure 6.18: *File Transfer Detected Monitoring On Guard*

Finally, with the transfer complete, **osdpspy** displays the content of the file that was transferred, as shown in Figure 6.18

6.4.7 Summary of Findings

Through our analysis of the Lenel X2210 access control unit we have found that:

- **osdpspy** performed as expected.
- The SCBK is generated by the access control unit. No user intervention is required other than to enable secure channel. Further research is required to determine how the SCBK is generated by the controller.
- We successfully decrypted the card data.

- We successfully decrypted PIN data.
- We successfully detected a valid access and revealed the card number and the corresponding PIN number.
- We successfully captured a file transfer and revealed the data contained therein.

6.5 Conclusion

In this chapter we have successfully skimmed sensitive data from two commercially available access control systems in laboratory conditions. We have also demonstrated that **osdpspy** is a general purpose research tool for analysing the OSDP protocol.

Chapter 7

Discussion

7.1 Introduction

In the previous chapters, we have built a prototype software application called **osdpspy** that can capture the initial key exchange in the OSDP protocol and then decrypt the ongoing traffic in-real time. We have seen how sensitive data may be gathered from the system as actionable intelligence using **osdpspy**. In this chapter we explore the implications of these results for the protocol and those that use it.

7.2 The Vulnerability

In earlier chapters, we have highlighted the vulnerability of using the default secure channel base key (SCBK) by the access control unit to share a private key with the peripheral device.

In Figure 7.1, we show the **osdpspy** output for the use of the default SCBK to set up a secure channel to the peripheral device. Note that **osdpspy** reveals the "secret" key that has been shared between the access control unit and the peripheral device.

Once the secure channel base key is known, it is saved by **osdpspy** for the next time an **osdp_CHLNG** command is issued by the access control unit. Each time

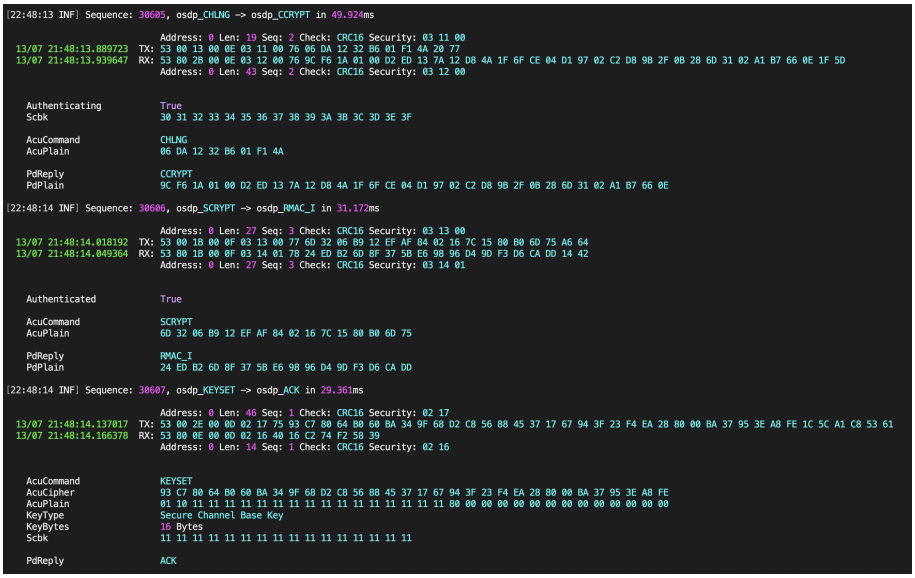

Figure 7.1: Secure Channel Setup Using Default SCBK

the secure channel is setup, **osdpspy** is able to generate the session keys, track the communication using the MAC values and decrypt the payloads of both commands and responses, where present. With the payloads now in plain text, we have shown how sensitive data is leaked and turned into actionable intelligence.

7.2.1 Security vs. Interoperability

In studying OSDP, we find that there is a tension between the security of the protocol and the interoperability of devices.

7.2.1.1 Security

Version 2.2 of the OSDP specification (Security Industry Association, 2020) states the following about the **osdp_KEYSET** command:

> *The command shall be sent by the ACU and accepted by the PD only while the connection is "secure". The "secure" context in this context shall mean that either:*

a) the current connection is encrypted, and the sessions keys are based on the current SCBK (or SCBK-D), or

b) that the connection is inherently secure via physical security, such as ACU/PD are connected via simple short cable. The "inherently secure" connection shall be asserted to the ACU and to the PD by setting the device into a special installation setup mode. The devices should exit the setup mode automatically after successful completion of this osdp_KEYSET command.

We argue that use of the default secure channel base key (SCBK-D) is NOT "secure" according to this definition.

7.2.1.2 Interoperability

The benefit of the default secure channel base key (SCBK-D) is the interoperability of devices. By using this method, manufacturers of OSDP devices can assure interoperability of the secure channel function with devices from other manufacturers.

We argue, however, that interoperability has been prioritised over security in the specification of the protocol.

7.2.1.3 Other Methods of Sharing the SCBK

Other methods could be employed to share the secure channel base key between the access control unit and the peripheral device, including:

- **Configuration Card** - The key is read by the peripheral device from a cryptographically protected RFID card, requiring a means to configure the key in the controller.
- **Manufacturer-Specific Command** - The key is passed using a special command requiring another shared key.
- **OSDP File Transfer** - The key is passed in an encrypted file using OSDP file transfer, requiring another shared key.

In each of these cases, the peripheral device implementation is specific to the manufacturer and the access control unit manufacturer would require knowledge of each implementation to support the respective key sharing methods.

We argue that none of these approaches serve the requirement for the interoperability of devices and only adds another layer of complexity to the implementation and deployment of OSDP in the field.

7.3 Potential Exploits

In this section, we examine how the vulnerability we have identified in Section 7.2 may be exploited by a bad actor to gain access through a door whose devices are utilising OSDP secure channel.

7.3.1 Deployment

It is possible to configure a Raspberry Pi to run Ubuntu and **osdpspy**. The Raspberry Pi is installed in a covert location tapping into the OSDP bus. WIFI access allows the Raspberry Pi to connect to the internet and log the data captured in real-time. Alternatively, the data could be captured locally and retrieved at a later date.

Installation of the device would probably require the cooperation, however induced, of an installer supporting the system to be attacked.

7.3.2 Actionable Intelligence

As we have seen in the description of **osdpspy** and the results of our analysis of two actual systems, the software can provide the following actionable intelligence:

- A record of a valid access at a door.
- The card number corresponding to the valid access.
- If required, the PIN number used to gain entry at the door.

Based on this data, it is possible to attempt various attacks on the system which will shall explore in the following sections.

7.3.3 Obtain Physical Card

If we can procure the use of a physical access control card through social engineering (e.g. borrow the card to go to the toilet) and the card number has been captured by **osdpspy** then we will be able to use the card anywhere the user is valid, even if a PIN is required for entry since **osdpspy** will also have provided the corresponding PIN number.

Borrowed Card + PIN = Valid Access

7.3.4 Clone A Card

A number of legacy card technologies can be cloned easily. If the card number and Wiegand format are known, it may not be necessary to physically access the card we wish to clone. Whilst it may be considered bad practice to deploy OSDP secure channel without upgrading the users' cards, this situation arises when a large organisation is migrating its users from insecure legacy credentials to a modern solution like DESFire EV2.

Card Number + Wiegand Format = Cloned Card

Cloned Card + PIN = Valid Access

7.3.5 Towards A Man-In-The-Middle Attack

In the previous two examples, we have required a physical card to gain access through the door. What if it were possible to remotely control the Raspberry Pi to play a valid card number and PIN to the system?

The Raspberry Pi deployment of **osdpspy** lays the foundation for a man-in-the-middle attack of OSDP and is a topic for further study.

The principle of the attack is as follows:

- **Phase 1** - Monitor the bus as normal and learn the characteristics of the access control unit and reader needed to successfully mimic their operation.
- **Phase 2** - Switch the Raspberry Pi inline between the access control unit and the reader (probably using relays). In this mode, the Raspberry Pi mimics the reader to the access control unit while passing data through from the reader.
- **Phase 3** - Access the Raspberry Pi remotely from a mobile phone and find out the valid cards and PIN numbers that have been used to access the door. Select one and request access.

osdpspy + Remote Access = Valid Access

In this section we have summarised the vulnerability of using the default secure channel base key (SCBK-D) in setting up the secure channel and some potential exploits for this vulnerability.

7.4 The Business Context

No discussion of OSDP would be complete without considering the market in which OSDP is promoted and deployed as the secure alternative to Wiegand. Throughout the supply chain, various promises are being made to the end users about the benefits of OSDP. A typical example can be found in an article in Security Magazine (Nemorofsky, 2020).

OSDP is only one current impetus to upgrade card and reader technologies in enterprise access control systems. The other impetus, as identified in Section 3.2.1.8 is to upgrade insecure card technologies. Low frequency card technologies have been known to be insecure for many years and there are a number of 13.56MHz technologies that have also been compromised including Mifare Classic, Legic Prime and iClass. Up to date information about the vulnerability of access control RFID cards can be found at **proxmark.org**.

We argue that a successful upgrade strategy for access control readers and cards includes approaches for both the communication between the reader and access control unit and also between the access control card and the reader.

The promise of OSDP starts with the Security Industry Association and the specification. We now examine the how the different actors in the supply chain view this promise.

7.4.1 Security Industry Association

The Security Industry Association (SIA) owns the protocol and has established it as an international standard: IEC60839-11-5:2020 (IEC, 2020). This was a major achievement as it was previously seen as an American standard and not generally taken seriously in Europe where proprietary encrypted protocols were already in use.

SIA offers the OSDP Verified (Security Industry Association, 2020a) programme for manufacturers to validate their products and has been in existence for over 12 months. How the programme is viewed varies from manufacturer to manufacturer (see below).

SIA also encourages all participants in the OSDP supply chain to contribute to the OSDP Working Group. The working group considers all matters related to the protocol from minor clarifications to the formulation of major new features.

7.4.2 Manufacturers

From the experience of Third Millennium engaging with various manufacturers, we can categorise their approaches as follows:

- **Engaged** - These manufacturers are engaged with the OSDP working group and their thinking and, in some cases, products are ahead of the current version of the specification. Many will also have already submitted their products to the OSDP Verified programme.
- **Implement and Move On** - These manufacturers implement their own interpretation of the specification with limited interoperability testing and, when challenged, will use their interpretation of the specification as a defence. Some of these manufacturers have also achieved the OSDP Verified mark.

- **Minimum Implementation** - These manufactures implement the minimum features needed to support OSDP. These products quite often omit secure channel.

7.4.3 Installers

Certainly in the UK, readers have been installed by electricians with little or no IoT skills. The reader is connected up to a controller and a card presented. If the reader goes beep then it can be assumed that the reader is working.

This is not the case installing an OSDP reader. As we have seen previously, additional steps need to be taken to pair the reader with the controller and initiate the secure channel. Until this has occurred, the reader will not respond to a card being presented.

This leads to more support calls to the manufacturers and the perception is that the readers do not work when, in fact, it is a system configuration concern.

7.4.4 End Users

End users are driven by cybersecurity concerns across their entire IT estate. OSDP is outside the scope of experience for many cybersecurity professionals making it hard to assess and manage the risks associated with its deployment.

OSDP is promoted as the secure alternative to Wiegand and, notwithstanding the vulnerability identified in this dissertation, a secure channel deployment is significantly better than a Wiegand deployment. End users are told that the link uses AES 128-bit encryption and this may be enough to check the box for security for many. Some may look for the OSDP Verified mark from their access control unit and reader vendors.

7.5 A Practical Mitigation?

We have demonstrated that if the reader and access control unit are paired for secure channel using the default secure channel base key, and if the communi-

cation is being monitored, it is possible to decrypt the secure channel communication.

The simplest practical mitigation to this problem is pair the reader and access control unit prior to installation. Pairing occurs in a controlled environment with the new secure channel base key generated by the controller.

We argue that this approach is only practical where small numbers of reader and access control units are involved. We do not believe this is a practical approach where hundreds of readers and access control units require pairing in a major enterprise deployment.

We also need to consider the case where a reader has failed (either genuinely or damaged by a bad actor) and how this reader gets paired with the access control unit. This will almost certainly happen in-situ and, once again, leave the communication path open to monitoring and the vulnerability we have described earlier.

7.6 Towards a Better Secure Channel

Whilst we do not propose a new method is this dissertation, we do argue that we, the OSDP community, do need to work towards a better secure channel. In this section we examine existing and new approaches that might help us achieve this in the future.

7.6.1 Diffie-Hellman

The fundamental problem that we need to solve to remove the vulnerability of the default secure channel base key is the exchange of a secret key over a public channel. This problem was solved in 1976 by the Diffie-Hellman key exchange (Diffie u.a., 1976) where public and private keys are generated to facilitate the transfer of the secret to be shared. The security of the exchange is provided by the computational cost of factoring two large prime numbers.

If the Diffie-Hellman key exchange protocol provides the foundational building block of internet security, then why was it not used in OSDP to set the secure channel base key?

At the time secure channel was being formulated, the majority of access control readers were implemented using severely, by today's standards, resource constrained 8-bit and 16-bit microcontrollers. These devices did not have the computational power or resources to perform the mathematical operations required by the Diffie-Hellman protocol.

Instead, the current method was chosen based on Secure Channel Protocol '03' (Global Platform Inc., 2014).

7.6.2 NIST Lightweight Cryptography

The Internet-of-Things presents a new resource-constrained computing challenge as recognised by NIST and their current Lightweight Cryptography project (NIST, 2021). This project seeks to balance security, performance and resource requirments for resource-constrained devices. This set of requirements is a good fit for an OSDP peripheral device.

7.6.3 Post-Quantum Cryptography

Given that future OSDP devices may be based on new, lightweight cryptographic methods, if is also important that any new methods are also quantum-safe. By quantum-safe, we mean that the method is thought to be secure against a cryptanalytic attack by a quantum computer.

In a world where powerful quantum computers exist (National Cyber Security Centre, 2016), Diffie-Hellman is no longer considered secure and new methods that solve the problem of sharing a secret over a public channel are actively being developed.

7.6.4 Building a New Method for Sharing the SCBK

The secure channel base key (SCBK) is the shared secret that secures communication between the access control unit and the peripheral device. We seek to share this secret over a public channel.

The requirements for a building a new secure channel session method might be framed as follows:

- The implementation must run efficiently on a 16-bit microcontroller
- The algorithms employed shall be "quantum-secure" according to NIST
- The session shall be setup using replacements for the **osdp_CHLNG** and **osdp_SCRYPT** commands

Of course, the largest challenge to building a new secure channel for OSDP is not solving the technical problems we present here. The real challenge is to promote a proof-of-concept to the OSDP community and working group and get a commitment to adopt the proposal and incorporate it into the OSDP specification.

7.7 Applications of osdpspy

osdpspy was born out of a curiosity about a set of data that was received. The application continues to evolve and finds uses in many settings.

7.7.1 Security Research

The prime use of **osdpspy** throughout this dissertation is as a security research tool. We have been able to decrypt the secure channel in real-time and capture sensitive data from the OSDP bus.

7.7.2 System Debugging

An invaluable role for the tool at Third Millennium has been to witness the secure channel operation of third-party products and diagnose integration and interoperability problems with those third-party products and also Third Millennium products.

It is very difficult to diagnose secure channel operation issues without a decrypted trace of the communication between devices.

7.7.3 Education

By showing newcomers a comprehensive trace from **osdpspy**, we can quickly build their knowledge about the protocol and best practice implementations for access control units

7.7.4 Security Sensor

An interesting potential application of **osdpspy** is as a sensor gathering and enriching information from the OSDP bus and feeding it into an anomaly detection system. We can also use the eavesdropping technique to alert a central monitoring station that unauthorised pairing of reader and access control unit have been detected and use the exploit in a positive, proactive manner.

7.8 Conclusion

In this chapter, we have summarised the vulnerability of the OSDP secure channel and examined some potential exploits. We have discussed how the OSDP supply chain adds complexity through vendor practices and consumer expectations. Finally, we explored the versatility of **osdpspy** and its myriad of applications.

Chapter 8

Conclusion

In this dissertation, we have developed a tool called **osdpspy** that can monitor and, under certain circumstances, decrypt the communication between an access control reader and an access control unit and provide actionable intelligence to the attacker of the system. In this chapter, we consider the implications of our work for the Open Supervised Device Protocol, its impact on embedded development at Third Millennium and the future of **osdpspy**. We also reflect up the time spent at Cardiff University culminating in this dissertation.

8.1 Implications for OSDP

Third Millennium continues to participate in the OSDP Working Group and we believe there is recognition among some members that a better secure channel is needed. As noted in Section 7.6.4, the real challenge is in getting to community to accept that there is a problem and collectively work towards a solution. This is so different to the approach taken at Third Millennium where a small team evaluate a problem its potential solutions and then implement at a rapid pace. Moving the OSDP Working Group to a new position requires a lot of effort, time and frustration.

In order to mitigate the vulnerability highlighted by this work, installers must be given the necessary tools and educated to pair readers offline and away from potentially prying "eyes". **osdpspy** has a role in educating all levels of practitioners about the protocol in general and secure channel in particular. This

potentially helps those installing OSDP devices to clearly understand the implications of pairing readers with controllers in situ.

Of course, the real threat to OSDP is from the IoT explosion currently pervading all aspects of embedded controller applications. IoT devices have some form of internet connectivity and therefore a TLS 1.3 implementation removing the need for an application-specific secure channel. Does this mean that OSDP will be swept aside in this revolution? Given the efforts across industry to move from Wiegand to OSDP in the last decade, the answer is almost certainly **NO** and it may take another decade to build industry-wide standards for the connection of access controls readers to a system. The available IoT products and standards do, however, provide a rich platform for the innovation of new access control devices that will, without doubt, continue to appear on the market.

8.2 Impact on Third Millennium Development

The development of **osdpspy** has led to further innovation at Third Millennium having a positive impact on product development.

One of the key challenges of developing an access control reader is to be able to monitor activity within the reader software. The device is constrained in terms of program flash memory, RAM and I/O. From our experience, the best strategy is to test and observe the behaviour of the reader. Traditionally, this has been achieved by evaluating the response to a given input. When the response is incorrect, we have to evaluate and debug the problem.

As the development of **osdpspy** proceeded, it became clear that the techniques used to capture and annotate the protocol could be applied to other communication paths within the reader. A Third Millennium reader has paths from the microcontroller to a bluetooth module, an RFID module and a keypad. In order to observe these protocols, we developed a protocol wiretap using a a microcontroller development board and were able to eavesdrop the communication between the microcontroller and these peripherals. The relevant protocol data is then fed into a protocol-specific application similar to **osdpspy**.

This technique has improved the observability of the reader sub-systems and through development of the relevant **spy** applications we plan to identify non-compliant behaviour earlier in the development cycle.

8.3 The Future of osdpspy

osdpspy is by no means complete. The objective of this dissertation was to decrypt secure channel communication, decode card and PIN data and to monitor the LED state of the reader in order to gather actionable intelligence about the system. There are numerous commands and responses in the protocol that do not yet have decoders to reveal the content of these messages. The tool may be considered "protocol complete" once the decoders have caught up with the current content of the OSDP specification.

To simplify the integration of other OSDP tools that need a logging interface, it is intended to add a TCP/IP socket interface to the product so that frames can be pushed to **osdpspy** from another application. This application would most likely be an access control unit. In addition to adding another input path, we also intend to expand the structured logging output capability to include Splunk. Splunk is a data capture tool commonly used for security logging and analysis.

The longer term plan is to open source **osdpspy** and make the tool extensible through plug-in libraries so that others may write their own annotators to view the protocol data to their own taste, or to present manufacturer-specific data. With an extensible version, Third Millennium would be able to add custom decoders based on the proprietary Wiegand formats Third Millennium has collected over the past three decades.

8.4 Personal Reflection and Final Words

I started thinking about studying for an MSc in a computer science subject around 2014 when it became clear that i needed a refresh with a particular interest in security and cloud computing. The final push came when my son started to apply for a university place. He is now studying for an MPhys at York while I complete my MSc Cybersecurity at Cardiff.

The first thing I noticed was how young the majority of my fellow students were and the variety of nations represented in class. I made new friends and had both the pain and joy of group assignments. Given my daily workload, I run a design and manufacturing business as well, I knew group assignments were going to be a pain point. I was not disappointed by the first assignment where

two of us did all the work without any real input from the other three until two days to go where they complained that they had not had any input into the process! Having survived the urge to withdraw from the course because of this experience, it was with a sense of foreboding that I came to the second group assignment. What a difference! Suddenly, here was a proper team with clear division of labour and a sense of fun. With this very positive experience under my belt, I proceeded confidently to the end of the taught part of the course.

What was clear throughout the course were the gaps in my "undergraduate" computer science knowledge. Even though I have been in industry for over 30 years, there were a number of undergraduate topics taught today that were not taught when I was an undergraduate. This was easily remedied by closing the gap through appropriate reading around topics where I was unclear or "rusty", but this consumed a lot of extra time.

The real joy of this course of study, though, has come from preparing this dissertation, by allowing myself to be properly immersed in an issue current to the security industry. The outcome of this work will hopefully allow me to impact the future of the Open Supervised Device Protocol, how it is deployed and the training of those who deploy it.

References

Baseggio, Mark / Evenchick, Eric (2015): *Breaking Access Controls with BLE Key*
Available at: https://www.blackhat.com/docs/us-15/materials/us-15-Evenchick-Breaking-Access-Controls-With-BLEKey-wp.pdf, Accessed on: 5th October 2021.

Datalust (2021): *Seq: Machine data, for humans*
Available at: https://datalust.co/seq, Accessed on: 5th October 2021.

Davis, Michael L. (2011): *Brushing Up on Wiegand: The man, the effect and the wire that changed engineering*
Available at: https://www.google.com/urlsa=t&rct=j&q=&esrc=s&source=web&cd=&cad=rjauact=8ved=2ahUKEwjD79aF7bPzAhWO_KQKHYItDI8QFnoECAoQAQ&url=https://www.machinedesign.com/Flearning-resources/Fengineering-essentials
Accessed on: 5th October 2021.

Diffie, Whitfield / Hellman, Martin E. (1976): *New Directions in Cryptography*
Communications of the ACM.

Elasticsearch (2021): *(Don't) let it go with the frozen tier*
Available at: https://www.elastic.co, Accessed on: 4th July 2021.

Farpointe (2020): *Magnetic Stripe Data Format*
Available at: https://www.securitytechnologiesgroup.co.uk/downloads/Ref_Pyramid_Series_Magnetic_Stripe_Data_Format.pdf, Accessed on: 27th October 2020.

Global Platform Inc. (2014): *Secure Channel Protocol '03' - Amendment D v1.1.1*
Available at: https://globalplatform.org/wp-content/uploads/2014/07/GPC_2.2_D_SCP03_v1.1.1.pdf, Accessed on: 4th November 2020.

Hacker Warehouse (2020): *BLEKey*
Available at: https://hackerwarehouse.com/product/blekey/, Accessed on: 27th October 2020.

Help Net Security (2007): *HID Global statement on IOActive withdrawing their Black Hat presentation*
Available at: https://www.helpnetsecurity.com/2007/02/28/hid-global-statement-on-ioactive-withdrawing-their-black-hat-presentation/, Accessed on: 5th October 2021.

HID Corporation (2006): *Understanding Card Data Formats*
Available at: https://www.hidglobal.com/sites/default/files/hid-understanding_card_data_formats-wp-en.pdf, Accessed on: 27th October 2020.

HID Global (2020): *Corporate 1000 Program*
Available at: https://www.hidglobal.com/access-control/corporate-1000-program, Accessed on: 27th October 2020.

IEC (2020): *IEC 60839-11-5:2020. Alarm and electronic security systems - Part 11-5: Electronic access control systems - Open supervised device protocol (OSDP).* , International Electrotechnical Commission.

LenelS2 (2020): *OnGuard*
Available at: https://www.lenel.com/products/onguard, Accessed on 25th October 2020.

McMaster, Nate (2021): *CommandLineUtils*
Available at: https://github.com/natemcmaster/CommandLineUtils, Accessed on: 4th July 2021.

Mehl, Bernhard (2018): *Hacking HID with Wiegand Protocol Vulnerability*
Available at: https://www.getkisi.com/blog/hid-keycard-readers-hacked-using-wiegand-protocol-vulnerability, Accessed On: 31st October 2020.

Microsoft Corporation (2021): *Download .NET*
Available at: https://dotnet.microsoft.com/download, Accessed on: 20th May 2021.

modbus.org (2006): *MODBUS over Serial Line: Specification and Implementation Guide V1.02*
Available at: https://modbus.org/docs/Modbus_over_serial_line_V1_02.pdf, Accessed on: 30th September 2021.

MVN Repository (2021): *What's New in Maven*
Available at: https://mvnrepository.com, Accessed on: 25th May 2021.

National Cyber Security Centre (2016): *Quantum-safe cryptography*
Available at: https://www.ncsc.gov.uk/whitepaper/quantum-safe-cryptography, Accessed on: 22nd August 2021.

Nemorofsky, John (2020): *OSDP: The future of access control*
Available at: https://www.securitymagazine.com/articles/92828-osdp-the-future-of-access-control, Accessed on: 22nd August 2021.

NIST (2021): *Lightweight Cryptography*
Available at: https://csrc.nist.gov/projects/lightweight-cryptography, Accessed on: 22nd August 2021.

Nuget.org (2021): *Create .NET apps faster with NuGet*
Available at: https://www.nuget.org, Accessed on: 20th May 2021.

OSDP Connect (2016): *How OSDP Was Developed*
Available at: http://www.osdp-connect.com/how-osdp-was-developed.html#, Accessed on: 31st October 2020.

proxmark.org (2013): *T55x7 and Tags Emulation*
Available at: http://proxmark.org/forum/viewtopic.php?id=1767, Accessed on: 27th October 2020.

Python Software Foundation (2021): *Find, install and publish Python packages with the Python Package Index*
Available at: https://pypi.org, Accessed on: 25th May 2021.

Rinaldi, John S. (2016): *Modbus Security*
Available at: https://www.rtautomation.com/rtas-blog/modbus-security/, Accessed on: 30th September 2021.

Sabt, M. / Traoré, J. (2016): *Cryptanalysis of GlobalPlatform Secure Channel Protocols*. Security StandardisationResearch, volume 10074 of LNCS, pages 62–9.Springer International Publishing.

Security Industry Association (2020): *Open Supervised Device Protocol (OSDP)*
Available at: https://www.securityindustry.org/industry-standards/open-supervised-device-protocol/, Accessed on: 1st November 2020.

Security Industry Association (2020a): *SIA OSDP Verified: Comprehensive Testing to Ensure Interoperablility*
Available at: https://www.securityindustry.org/industry-standards/open-supervised-device-protocol/sia-osdp-verified/, Accessed on: 1st November 2020.

Security Industry Association (2021): *osdpcap - libosdp-conformance OSDP capture format*
Available at: https://github.com/Security-Industry-Association/libosdp-conformance/blob/master/doc/osdpcap-format.pdf, Accessed on: 16th May 2021.

Serilog (2021): *Flexible, structured events - log file convenience*
Available at: https://serilog.net, Accessed on: 24th May 2021.

Software House (2020): *C-CURE 9000 Security + Event Management*
Available at: https://www.swhouse.com/products/CCURE_9000.aspx, Accessed on 25th October 2020.

Splunk (2021): *The Data-to-Everything Platform, Powering Security, IT and DevOps*
Available at: https://www.splunk.com, Accessed on: 4th July 2021.

Third Millennium (2020): *Advanced Access Control Readers*
Available at: https://www.tm-readers.com, Accessed on: 26th October 2020.

Wehr, John (2003): *The Wiegand Effect: The 30-year old science project still influences modern security systems*
Secure Id News, Available at: https://web.archive.org/web/20180412142831/https://www.secureidnews.com/news-item/the-weigand-effect-the-30-year-old-science-project-still-influences-modern-security-systems/, Accessed On: 26th October 2020.

ZDNet (2007): *Nixed: Black Hat talk on RFID access badge risks*
Available at: https://www.zdnet.com/article/nixed-black-hat-talk-on-rfid-access-badge-risks/, Accessed on: 27th October 2020.