ST. PÖLTEN UNIVERSITY
OF APPLIED SCIENCES

Informatik
& Security

# Air-Gapped Network: Bridging the Security Gap

Master thesis

for attainment of the academic degree of

Master of Science in Engineering (MSc)

submitted by

SAURABH DAVE

52304918

in the

University Course Cyber Security and Resilience at St. Pölten University of Applied Sciences

Supervision

Advisor: FH-Prof. Dipl.-Ing. Dipl.-Ing. Christoph Lang-Muhr, BSc

Assistance: -

# Declaration of Honour

First name, Surname: SAURABH DAVE

Matriculation number: 52304918

Title of the thesis: Air-Gapped Network: Bridging the Security Gap

I hereby declare that

- I have written the work at hand on my own without help from others and I have used no other resources and tools than the ones acknowledged
- I have complied with the Standards of good scientific practice in accordance with the St. Pölten UAS' Guidelines for Scientific Work when writing this work.
- I have neither published nor submitted the work at hand to another higher education institution for assessment or in any other form as examination work.

Regarding the use of generative artificial intelligence tools such as chatbots, image generators, programming applications, paraphrasing and translation tools, I declare that

☐ no generative artificial intelligence tools were used in the course of this work.

☒ I have used generative artificial intelligence tools to proof-read this work.

☐ I have used generative artificial intelligence tools to create parts of the content of this work. I certify that I have cited the original source of any generated content. The generative artificial intelligence tools that I used are acknowledged at the respective positions in the text.

Having read and understood the St. Pölten UAS' Guidelines for Scientific Work, I am aware of the consequences of a dishonest declaration.

# Abstract

Air-gapped networks is used as security mechanisms that protect sensitive systems and classified information through physical or logical separation from vulnerable networks like the Internet or corporate LANs. Due to the lack of external communication links, these networks offer limited opportunities for attackers to gain access. Critical infrastructure systems like Military and Government Systems protect vital data by utilizing air-gapped environments as essential security measures.

Air-gapped networks provide robust protection from unauthorized access and malware, prevent data breaches in today's cybersecurity environment, where attack threats increase in size and complexity. Taking away direct network connections leads to a substantial reduction in attack surfaces while protecting systems from current cyber threats. Operational requirements along with business needs frequently demand secure data movement between air-gapped systems and online networks.

This thesis introduces a controlled and secure approach to exchange data between isolated and networked systems through a data diode implemented in a layered security framework. The system architecture guarantees that both data confidentiality and integrity remain intact throughout the transfer process. The system implements strict access control to permit only authorized users to start data transfers while also establishing policies for allowable data types and keeping detailed logs for comprehensive auditability of data movements.

Traditional transport-layer acknowledgments cannot operate since the data diodes support only unidirectional communication. The thesis proposes an application level acknowledgment mechanism which ensures data transfer reliability and traceability to overcome existing limitations. The solution achieves a balance between security requirements and operational functionality to create a practical system for environments that need robust protection and uninterrupted operations.

**Keywords**: Air-Gapped, Secure Data Exchange, Air Gap, Cross-domain communication

# Contents

# 1. Introduction

Over the past few years, as internet dependence has developed, cyberattacks have emerged in various forms, including hacking and malware distribution. Intelligent and persistent forms of cyber threats have been damaging to the organizations' cyber assets and missions. These attacks have levied collateral damage on both Nations and Organizations by leaking sensitive information, disrupting systems and compromising websites and networks [1]. The Microsoft Digital Defense Report 2024 highlights a significant increase in Cyber Threats with Nation-State Cyber Operations, rise in Cybercrime and Ransomware attacks, explosion in Social Engineering, growing Distributed Denial of Service (DDoS) and Artificial Intelligence (AI)-powered cyber threats becoming more sophisticated and widespread [2]. As per Google Cloud Security, Cyber Threats will evolve rapidly in 2025, driven by AI Advancements, Geopolitical Conflicts and Cyber Criminal innovations. Therefore Organizations must strengthen their defenses, prepare for Post-Quantum Cryptography challenges and take Proactive Cybersecurity Measures [3].

The rising complexity of cyber threats pushes organizations to demand more secure network architectures. Air-gapped networks have emerged as a key defensive mechanism because of their exceptional robustness. Air-gapped networks maintain their security by remaining physically separated from external networks such as the public internet and insecure networks. This design produces an isolated environment that keeps essential systems entirely separate from the connected networks which attackers might access. Air-gapped systems protect against cyber espionage and unauthorized data access by removing remote access capabilities which helps prevent Advanced Persistent Threat (APT).

High-security organizations like government agencies, military installations, financial institutions, and industrial control systems commonly use air-gapped environments to protect their sensitive operations [4]. Operational data security remains critical in high-risk settings because breaches result in extensive negative effects. While air-gapping delivers superior security advantages its implementation faces challenges since operational demands necessitate regular data exchanges between segregated systems and networked environments. The need to perform software updates, operational data analysis or dataset sharing with approved external parties creates these requirements.

Historically, organizations transferred data to and from air-gapped systems manually by using USB drives or other removable media which authorized staff handled [5]. While simple to use, this method introduces significant security vulnerabilities and operational limitations. Manual data transfer methods are both resource-heavy and error-prone while remaining vulnerable to internal threats and physical interference. Manual methods face scaling challenges during deployment in large enterprises due to increased frequency and complexity of data exchange requirements.

The task of connecting isolated air-gapped systems to networked environments now represents a crucial research focus in cybersecurity. Secure data transfer between isolated systems requires automated mechanisms which adhere to strict policies to protect system integrity, confidentiality and availability while exchanging classified data. The solution to this challenge requires choosing suitable technologies while establishing strict security policies and implementing defense in depth through multi-layered architecture together with the adoption of best practices designed for high-assurance environments.

Multiple hidden approaches exist for data transfer between an air-gapped network and a connected network. These methods function by utilizing sound, light, heat, or electromagnetic signals to overcome network isolation [1] [5]. The use of these methods in unauthorized attacks makes them security threats which exclude them from legitimate communication channels and thesis scope.

## 1.1. Research Questions

- How to facilitate secure and controlled data exchange between air-gapped network and connected network?
- What mechanisms ensure that authorized users can transfer approved data types only?
- How to ensure accountability and attribution of data transfer?
- How to protect air-gapped network from incoming traffic from connected network?
- How to acknowledge transfer status back to initiating network?

## 1.2. Contribution

This thesis makes the following contributions:

- Proposal of a data exchange methodology between air-gapped and connected networks using a data diode, leveraging a multi-layered architectural approach.
- Demonstration of controlled data exchange mechanisms, ensuring that data transfers are restricted exclusively to authorized users, limited to approved data types, and fully auditable.

- Presentation of techniques to maintain confidentiality and integrity of data after its transfer to a lower-security network environment.
- Development of a protection mechanism for air-gapped networks, employing a data sanitization approach to mitigate risks associated with incoming data.
- Provision of a solution addressing application-level acknowledgment of transfer completion, effectively overcoming the inherent limitation of one-way communication imposed by the data diode.

## 1.3. Thesis Outline

The overall structure of this document is as follows:

- **Introduction**: As discussed earlier, the introduction (referenced as chapter 1) provides an overview of the topic, challenges, motivation, and the scope of the research.
- **Background**: This section (referenced as chapter 2) covers necessary prerequisites and fundamental knowledge relevant to the subject.
- **Related Work**: The related work section (referenced as chapter 3) presents existing research and literature that pertains to our study.
- **Methodology**: In this part (referenced as chapter 4), we delve into the methodology employed in our research.
- **Implementation**: The implementation section (referenced as chapter 5) discusses the practical implementation of our approach.
- **Results**: In this section, we report the results of our experiments in the section labeled chapter 6.
- **Discussion**: The Discussion section (referenced as chapter 7), explores opportunities for enhancement, identifies areas where the current approach could be improved, and suggests modifications or future adaptations.
- **Conclusion**: Finally, chapter 8 summarizes key findings and contributions of this thesis, highlighting implications and recommendations for future research.

# 2. Background

An air-gapped network is a security measure used to protect highly sensitive systems by physically or logically isolating them from unsecured networks, including the Internet [5]. The Other term used for air-gapped network is Isolated network or high side network. Network connected to internet is called low side network. Air-gapped network is depicted in fig. 2.1. Red dotted line indicates the air gap between the network.

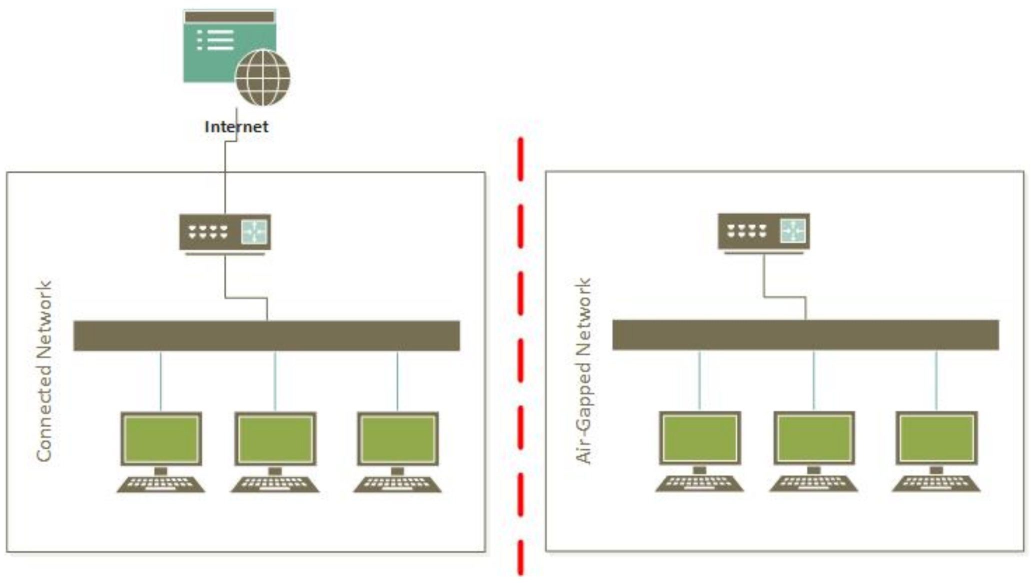

Figure 2.1.: Air-Gapped Network

## 2.1. Types of Air-Gapped Network

- **Physical Air-Gap Network**: A physical air-gap network represents an absolute physical separation between a system or network and all other networks including the internet and unprotected internal networks. The setup demands a total absence of wired and wireless links between the secure network and any other network. When required communication or data transfers take place they happen manually through physical media such as USB drives, CDs or external hard drives under strict security protocols [4] [5].

- **Logical Air Gap Network**: A logical air-gap network employs software-based and logical security measures rather than complete physical isolation. It creates a virtual separation or boundary through the use of technology such as firewalls, intrusion detection/prevention systems, data diodes (one-way data transfer devices), Virtual Private Network (VPN), or segmented Virtual Local Area Network (VLAN) [4] [5].

## 2.2. Use cases of Air-Gapped Network

The following are typical use cases for air-gapped networks:

- **Government and Military Systems**: The government and military agencies manage sensitive information which covers classified data and defense strategies. Critical systems are kept isolated from unsecured networks through air-gapped networks which serve as a barrier against unauthorized access and cyber threats for these organizations. Such physical separation substantially reduces the likelihood of espionage activities and both data breaches and cyberattacks [4] [5] [6].
- **Financial Institutions**: Banks and stock exchanges function as financial institutions that handle massive quantities of sensitive financial information while executing essential transactions. The use of air-gapped networks provides protection for financial assets against cyber threats while maintaining transaction integrity and data confidentiality. Institutions protect against financial fraud through the isolation of essential systems from unauthorized access [4] [5] [6].
- **Healthcare Industry**: In the healthcare sector confidential patient records together with medical research data require careful management. The use of air-gapped networks protect sensitive information. It also ensures adherence to the standards like the Health Insurance Portability and Accountability Act (HIPAA). The isolation method ensures patient confidentiality protection by preventing unauthorized access [5] [7].
- **Energy Sector**: Industrial Control System (ICS) has a vital role in managing operations of critical infrastructure such as nuclear power plants and water treatment facilities. The implementation of air-gapped networks makes critical systems secure from cyber threats while guaranteeing dependable operation of essential services. Separating industrial control systems from outside networks effectively reduces the chances of sabotage and unauthorized manipulation [5] [7].
- **Research and Development Institutions**: Research and development organizations manage both intellectual property and confidential proprietary information. By using air-gapped networks organizations protect sensitive research data from espionage activities and cyber threats thus maintaining

security for their innovations and projects [5] [7].

- **Legal and Law enforcement agencies**: Law enforcement agencies alongside legal firms handle confidential client information and sensitive case file documentation. Air-gapped networks secure legal documents by preventing unauthorized access and tampering which protects their integrity and confidentiality [5] [7].
- **High Security facilities**: Critical infrastructure along with sensitive data repositories operate within data centers and top-secret research labs. Air-gapped networks form secure boundaries that protect these systems from unauthorized access and external threats [7].

## 2.3. Advantages of Air-Gapped Network

- **Enhanced Security**: The Air-gapped networks provide strong security. Physical separation from external networks creates an extra protective layer for critical systems against cyber threats. Sensitive data cannot be accessed or compromised by attackers due to the absence of external network connections [5] [7].
- **Protection against Targeted Attacks**: The isolation of air-gapped networks delivers strong defense mechanisms against sophisticated cyber threats. The attack surface of these systems drops because they lack direct internet access which reduces the probability of successful intrusions and vulnerability exploits [5] [7].
- **Safeguarding Sensitive Information**: Air-gapped networks provide essential protection for sensitive information in government operations and military defense systems as well as financial institutions and healthcare facilities by maintaining data confidentiality. Physical data separation functions as a robust protection mechanism that prevents unauthorized access and maintains both privacy and security [5] [7].
- **Limiting Spread of Malware**: Air-gapped networks provide a strong protective layer that prevents malware and harmful software from accessing system networks. The spread of malware through isolated environments is limited due to their lack of direct external network connections which lowers infection risks and reduces ransomware threats and data loss events [7].
- **Reducing Vulnerabilities**: Air-gapped networks increase their security by eliminating outside connections which minimizes potential attack points and vulnerabilities. Computing systems that lack outside interfaces and software access experience significantly lower risks from cyberattacks and unauthorized access [5] [7].

- **Regulatory Compliance**: Organizations can fulfill cybersecurity and data protection regulation compliance through the use of air-gapped networks. Strictly regulated business sectors including finance and healthcare use air-gapped networks to demonstrate their commitment to data protection regulations and security accountability [5] [7].
- **Physical Security**: Physical security measures such as secure facilities and controlled equipment access along with surveillance systems form the foundation of air-gapped network security. These networks decrease the threat of unauthorized physical tampering or interference by allowing only authorized personnel to access them physically [7].

## 2.4. Challenges of Air-Gapped Network

Here are the challenges of air-gapped network:

- **Operational Complexity**: The establishment and management of air-gapped networks presents organizations with challenges that require significant resource investment. To achieve complete physical isolation dedicated hardware components and additional infrastructure resources are required along with detailed planning. Effective implementation and maintenance of isolated networks requires substantial organizational resources for proper execution and continuous monitoring [4] [7].
- **Limited Functionality**: Air-gapped networks purposely eliminate external connections which makes some tasks more difficult to manage. To move data between isolated networks and external systems organizations must use manual approaches including the use of USB drives and direct equipment connections. Manual operational procedures reduce efficiency and generate extra complexities within everyday business activities [7] [6].
- **Insider Threats**: Air-gapped networks provide strong defense mechanisms against outside cyberattacks yet insider threats continue to pose significant risks. Network security risks arise when individuals with physical access act maliciously or negligently. Organizations must implement rigorous access controls alongside ongoing monitoring and extensive security training for employees to mitigate these risks [4] [7].
- **Malware Propagation**: Air-gapped networks remain vulnerable to malware despite being isolated from direct internet access. Networks are vulnerable to malware infiltration through removable storage devices such as USB drives which are employed for data transfer. To maintain security organizations must implement strict policies and conduct thorough inspections together with regular malware scans for every removable media device [4] [7].

- **Usability Challenges**: Air-gapped networks present several practical difficulties because of their isolation from other systems. Because of their physical isolation air-gapped systems make routine maintenance tasks such as software updates and security patch installations more difficult to perform. The lack of internet connection prevents organizations from accessing cloud services and real-time threat intelligence which could restrict productivity [7].
- **Maintenance and Updates**: Regular maintenance and updates are essential to maintain the operational security of air-gapped networks. Organizations must promptly apply security patches and software updates while conducting continuous audits to maintain network security. The establishment of a secure isolated network environment demands significant effort together with proper resources and careful planning [7].
- **Human Error**: Human error represents the primary security risk to air-gapped networks. The chance for unauthorized entry exists when employees connect an infected USB drive by mistake or when they fail to secure network ports properly [4] [6].
- **Supply Chain Attacks**: Manufacturing or distribution stages of air gap networks may become attack vectors when malicious software or hardware is inserted. Network security remains at risk despite its robust safeguards because compromised components or devices can cause vulnerabilities [4].
- **Physical Attacks**: Air-gapped networks maintain physical separation but physical attacks still represent a security threat to them. If an intruder gains network access they can breach security through methods like hardware manipulation and physical safeguard circumvention along with social engineering approaches for unauthorized entry [4].

## 2.5. Real world Cyberattacks in Air-Gapped Systems

Air-gapped systems that are isolated from networks have been considered to be very secure. Yet there have been reports of such systems being breached. These breaches have shown to use unconventional means for communication also known as covert channels such as Acoustic, Electromagnetic, Magnetic, Electric, Optical, and Thermal to transfer data [5] [8].

### 2.5.1. Stuxnet

Stuxnet was first discovered in 2010. Stuxnet was an exceptionally complex malware which functioned as a highly advanced cyber-warfare tool developed to destroy industrial control systems with particular focus on Iran's Natanz nuclear facility. Stuxnet targeted industrial equipment to physically destroy it unlike

conventional cyberattacks which focus on confidentiality, integrity and availability.
Stuxnet infected Windows machines but specifically sought out Siemens industrial controllers attached to these machines with a focus on models S7-315 and S7-417. The malware executed a precise fingerprinting method to confirm controller configurations so that it affected only specific intended targets. Once Stuxnet found the targeted controllers it uploaded malicious code which secretly disrupted their operations. The malware actively controlled physical machinery including uranium enrichment centrifuges [9].

### 2.5.2. The Equation Group

The Air-Gap system attacked by the Equation Group was compromised using a specialized method known as "interdiction". In this method, attackers intercepted and replaced items (like CDs or USB sticks) during shipping with compromised versions. A specific example of such an attack included attendees at a scientific conference in Houston who, after the event, received CDs in the mail containing conference materials. These CDs were infected with malicious autorun software. The primary malware used for attacking air-gapped networks was named FANNY.
FANNY operated as a sophisticated worm specifically designed to map air-gapped networks. Once an infected USB was connected to the target air-gapped system, the malware identified network components, collected essential system information, and stored it in a hidden section of the USB. Later, when the infected USB was connected to a machine with internet access, it transmitted the collected data. The attackers used the USB's hidden storage to encode commands, executing them when the USB was reinserted into the air-gapped computer, allowing remote control and mapping of air-gapped networks [10].

### 2.5.3. NotPetya

NotPetya, released in 2017, was believed to be ransomware. NotPetya injects malicious codes in the computer and then attempts to gain administrator access. Following that, it infects other computers in the network. NotPetya uses the EternalBlue Server Message Block (SMB) exploit to conduct the attacks. The hard drives get encrypted, and when the computer is booted, the ransom note is displayed. NotPetya does not provide enough information for a decryption key to be produced, making it a malware. Businesses across industries have been affected without having an opportunity for system recovery[11].
On 27 June 2017, a wiper malware disabled numerous computers around the world, especially in Ukraine. It did this by deleting hard drives. The NotPetya malware entered a local network via a supply chain attack on the update mechanism of Ukraine's M.E.Doc tax management software. The malware spread independently, much like a worm, to companies that used the aforementioned software. Companies in numerous states were

infected with NotPetya. The attackers managed to infiltrate Ukrainian IT networks, systems of the National Bank of Ukraine, Kyiv Borispyl International Airport, the capital’s metro and the agency for managing the exclusion zone around the damaged nuclear power plant in Chernobyl [12].

# 3. Related Work

This chapter demonstrates the extensive effort dedicated to researching whether prior attempts have addressed similar problems or scenarios. Here, we document the state of the art as it was known at the time of writing, thereby highlighting the distinctiveness of our work in the subsequent chapters.

## 3.1. Secure One Way Data Transfer Technology

Data Diode is the technology to provide one-way communication channel and eliminate the possibility of using covert channels. Data diode provides one-way communication channel on physical level without possibility of transmission control in automatic mode. The main idea is that the source has just a transmission path on physical level and the receiver has just a reception path on physical level [13].

The data diode is secure by design and it has no documented evidence of flow being reversed. Network attacks can be classified in broad groups, namely: Infection, Exploding, Probe, Cheat, Traverse, and Concurrency [14].

To protect against the above listed and many other attacks, a variety of defense mechanisms have been proposed and used over time, like Routers, ACLs, Firewalls, and Intrusion Detection System (IDS) and Intrusion Prevention System (IPS). A common thing with all of the listed defense mechanisms is the presence of software modules that needs complex configuration. Most of the limitations of the above security tools stem from their software-based nature and which has required hardware-based security that doesn't run into configuration issues.

- Use Cases and Applications of data diodes [15]:
  - Enhancing IoT Device Security
  - Protecting Critical Infrastructure like Water, Oil, Gas, and Electric [16]
  - Protecting Nuclear power plants [17]
  - Defence and Military Communication
  - Mitigating IoT Threats
- Limitations of data diode [14]:

  - Not suitable for environments requiring two-way data exchange.
  - Costly and complex to implement, often limited to high-security or critical infrastructure (e.g., military, healthcare, nuclear).
  - Does not support real-time acknowledgment or error correction like TCP/IP naturally would.

The security provided by a data diode for transferring data between networks with varying security classifications does not suffice to ensure controlled data transfer on its own. Data transfer must involve mechanisms that permit only authorized users to begin transfers while limiting transfers to approved data types and including accountability and attribution measures. The implementation of a data diode does not fulfill all necessary conditions for controlled data transfer.

## 3.2. Cross Domain Solutions

A Cross Domain Problem (CDP) is the question of how to securely access and exchange information between the domains of varying security levels. A Cross Domain Solution (CDS) addresses the CDP by designing the framework and protocols for such access and transfers. Traditional CDS depend on trusted personnel and manual processes, which are costly and risky. The key challenge is enabling untrusted network gateways to process routing decisions without accessing sensitive data. Homomorphic Encryption (HE) provides the solution by allowing operations on encrypted data, maintaining confidentiality throughout processing [18]. The other solution is to use Functional Encryption (FE) for CDS [19].
The other approach to CDS is Data Guards. Traditional solutions rely on securing labels to decide if information can be transferred. Current method of making dirty word list is static and it has its own weaknesses. The proposed solution to this problem is machine learning based content checkers [20].

## 3.3. Content Disarm and Reconstruction

Content Disarm and Reconstruction (CDR) is a zero-trust file methodology that proactively extracts threat attack vectors from documents and media files [21]. CDR's ability to disarm and reconstruct relies on understanding the file format specification and being aware of its weakness. Therefore, CDR should handle each supported file type separately and validate the effectiveness. The CDR utilizes the in-depth understanding of the file structure to break down files into their discrete components. The CDR strips away anything that does not conform to the file type's original specification, International Organization for Standardization (ISO) standard, company policy, and rebuilds a "clean" version that has a high safety level against zero/one-day attacks. Since CDRs rely on understanding the file format, any CDR solution should handle each supported file type separately due to the vast difference in each format [22].

To prevent the attacks that spread via shared files that contains malware, the security industry responded with solutions that include tools that would strip the content out of a file and then reconstruct it into a new clean file. This is often called file sanitization. A file sanitization solution can be deployed at the entry point of the corporate network that look into a Word doc or spreadsheet and extract the words, cells, and formatting; then it builds a new clean file with the correct extension. There are few commercial vendors that have solutions for this [23].

# 4. Methodology

The study's methodology includes a thorough literature review and a comparative analysis to assess existing approaches. A systematic risk management and mitigation helps identify and tackle potential challenges tied to the research objectives.

## 4.1. Literature Review

To understand secure communication methods between networks with different security classifications I started by searching academic literature on platforms such as Google Scholar, IEEE Xplore, ACM, ResearchGate and other academic sources. I started my keyword research with terms like “high-side low-side network security” and “isolated network security” because I expected them to produce useful information about physically or logically separate systems.

My examination of initial sources demonstrated that scholarly and industry literature uses the term "air-gapped network" to refer to these architectural solutions. Finding the term "air-gapped network" became a pivotal moment in my research methodology which led to more focused and effective searches. I updated my search approach to focus on more precise key terms including “air-gapped network security”, “secure communication in air-gapped environments” and “air-gapped data exchange mechanisms”. The selected search terms delivered better quality literature results that were more relevant to my research.

Research in this field uncovered a wider domain of solutions which are collectively known as Cross Domain Solutions (CDS). The term commonly appeared in discussions about government operations and defense strategies as well as critical infrastructure projects which require protected data movement between isolated systems. Cross Domain Solutions (CDS) utilize both policy and technical mechanisms to enforce stringent controls between domains with different levels of sensitivity or trust.

The ongoing literature search uncovered multiple technologies that enable secure communication in air-gapped environments. The leading technologies in this domain include data diode technology which is a hardware-based unidirectional data transfer mechanism along with air fiber technologies that use wireless communication to connect air-gapped systems and maintain strict data flow control. These technolo-

gies, while varied in implementation, share a common objective: These technologies facilitate network-to-network data transfer while preserving each network's security integrity.
Through the literature review process I improved my technical landscape understanding and developed the conceptual framework that guided my research methodology. The transition from broad terminology to more precise language, coupled with the recognition of applicable technologies and methods provided a strong basis for subsequent design and implementation processes.

## 4.2. Comparative Analysis

Several methods for transferring data between air-gapped and connected networks were identified and analyzed for this thesis, including Sneakernet, One-way Cable Assembly, Firewall enabled policy, and One-way System Hardware (Data Diode) [24]. The final method for transferring data between the two networks was selected after carefully evaluating the advantages and disadvantages of each available option.
Sneakernet refers to transferring data by physically moving storage media (e.g., hard drives) from one computer or location to another. It is typically used when internet connections are unreliable or when transferring large data volumes quickly. Sneakernet is cost effective and provides high throughput, however it involves high latency, manual interventions and security risk from manual interventions [25].
One way cable assembly is modifying physical cable for one-way data flow. It is low cost and provides simplicity, it has very low throughput and is vulnerable to physical tampering [24].
A firewall depends on configuring code and policies set by admin. It has an inherent vulnerability that they cannot get rid of zero-day attacks and unknown new attacks. Some types of firewalls just depended on the head of the packet to permit or deny it [14].
Compared to the previously mentioned methods, the data diode is inherently secure by design, ensures strict one-way communication, and completely prevents any possibility of reverse traffic flow [13] [14] [15] [16] [17].

## 4.3. Risk Assessment and Mitigation

Once a data transfer method between air-gapped and connected networks is selected it is must to evaluate the possible threats tied to the transfer solution and establish methods to prevent or reduce these risks. The analysis has taken into account three primary risks which include:

1. **Unauthorized Data Transfers from Air-Gapped Network**: The one-way transmission feature of data diodes fails to automatically limit authorized users from transferring data that is not pre-approved.

To mitigate this risk organizations need to implement strict security policies and procedures that define allowed data types and enforce comprehensive compliance controls.

2. **Confidentiality Risks on the Connected (Low-Side) Network**: Sensitive or confidential information from the air-gapped network becomes at risk when processed or stored on the connected network with weaker security measures. Any data that moves past the secure network perimeter should undergo encryption with strong cryptographic methods to prevent unauthorized access to the information.
3. **Incoming Malicious Data Threats to Air-Gapped Network**: The security of an air-gapped network faces serious dangers when transferred data from a connected network includes harmful content. The network must incorporate strong defense systems like malware scanning and content checks together with strict inspection procedures to identify and block threats to protect the secure network from incoming data.

Together these mitigation strategies facilitate secure information transfer through controlled processes while protecting network integrity and confidentiality.

# 5. Approach

A multi-layered middleware application is designed to enable secure and asynchronous data transfer between an air-gapped network and a connected network. The system introduces the concept of channels, each defining "what" data can be transferred and "who" can initiate the transfer. To enhance security, users are not allowed to upload arbitrary files freely—instead, the application generates transfer files based on user input, ensuring data integrity and control. Process pipeline and order of the processes can be configured per channel which determines to complete the request for that particular channel which processes are called and in which order. Channels are further categorized based on approval workflows, allowing flexibility in how data is managed and governed. Each transfer operation includes an acknowledgement mechanism from the receiving side, confirming successful delivery and processing—ensuring end-to-end traceability and auditability. This architecture enforces defense-in-depth by combining structured controls, secure workflows, and extensive configurability.

**Key Features**

- **Channel-Based Architecture**: Logical channels isolate data transfer flows, with each channel managing specific configurations
- **Two Categories of Channels**:
  - **Pre-Approved Channels**:
    - No manual approval required
    - Only the application can initiate the request
    - File/Data, to be transferrred, are generated based on structured user input or selection within the application
  - **Approval-based Channels**:
    - Authorized users can initiate the request of arbitrary files
    - Each transfer requires manual approval, providing an extra layer of security for unstructured data
- **Controlled Transfer Initiation**: Transfer files are constructed by the application based on user inter-

actions, ensuring no arbitrary data is sent and reducing risk of data leakage

- **Asynchronous Transfer Mechanism**: Data is transferred in a asynchronous manner, enabling scalability, availability, reliability, and responsiveness
- **Acknowledgment on Delivery**: Status of the transfer request is acknowledge back to the sender side ensuring traceability across the entire transfer pipeline
- **Granular Access Control ("Who")**: Restricts access based on identity—either human users or authorized application services
- **Data Governance ("What")**: Defines the types of business data allowed per channel, tightly controlling sensitive and business-critical information
- **Per-Channel Configuration**: Each channel can be tailored with:
  - Approval requirements
  - Error notification settings
  - Custom validations and processing logic
- **Logging and Auditing**: Comprehensive logs are maintained for all transfer activities, including user actions, approvals, delivery acknowledgments, and exceptions that support security auditing.
- **Channel Extensibility**: Channels can be extended with custom workflows, data transformations, and other channel-specific logic to support unique business needs
- **Enhanced Security & Defense-in-Depth**: The system enforces strict control over every aspect of the data flow, combining access control, structured file generation, approval workflows, and acknowledgment-based completion tracking

## 5.1. Design Consideration

The middleware application primarily comprises two main components: the Sender Component and the Receiver Component. Each component includes several processes designed to facilitate secure and controlled data transfers between an air-gapped network and a connected network. Communication is strictly unidirectional at any given time, with independent mechanisms ensuring that bidirectional communication is entirely prohibited. High level overview of middleware application shown in fig. 5.1

The Sender and Receiver components are intentionally designed for reusability, allowing either component to function in both transfer directions, contingent upon their deployment scenario. Specifically, when transferring data from an air-gapped network to a connected network, the Sender Component is deployed within the air-gapped environment, and the Receiver Component is positioned within the connected network.

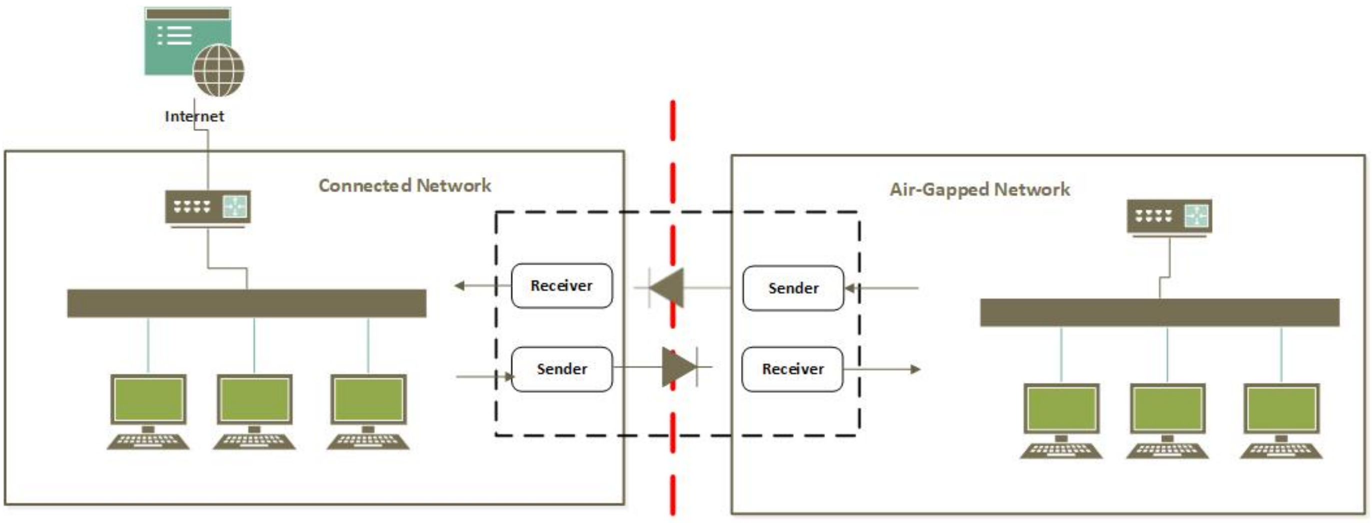

Figure 5.1.: File transfer mechanism between air-gapped and connected network

Conversely, for data transfers from a connected network to an air-gapped network, the Sender Component resides in the connected network while the Receiver Component operates within the air-gapped network.

The middleware facilitates the transfer of both files and text-based messages. Both the Sender and Receiver components comprise multiple processes, which can be independently configured for each channel. The total number of processes executed by each component is as follows:

**Sender Component Processes**:

- **File Upload**: The purpose of this process is to upload the file through REST API. This process returns unique identifier for the uploaded file
- **Transfer request initiation with metadata**: Once file is uploaded, consumer application can initiate the transfer by supplying other required parameters like unique identifier generated from the file upload process, requesting channel name, target users of the file, text message relevant to transferred file etc. This process is trigerred through REST API method call.
- **File Encryption**: This process encrypts the file for target users so that unauthorized users cannot access the content of the file.
- **Encrypt and Sign metadata**: The purpose of this process is to generate metadata which can be used to process and deliver the file to the other side of the network. The process then sign the metadata by service account used to run the process and encrypts the metadata for the target service account which receives the metadata.
- **Secure handover of files to the data diode for cross-network transfer**: This process combines encrypted file and encrypted and signed metadata and creates one package. This package is signed by

service account and handed over to data diode to for cross network transfer. Data diode transfers the package to other network only after successful verification of the signature.

- **Audit Log**: The purpose of this process is to record the start and end entries of each process for every request. These logs can be audited to ensure traceability of individual requests.
- **Routing Process**: This process routes messages from one process to another.
- **Retry mechanism**: Process makes pre-defined re-attempts in case if error or exception occurs during the processing of the file

**Receiver Component Processes**:

- **File reception from the data diode**: This process receives the package from the data diode and pass it on to the next process in pipeline for further processing.
- **Metadata Verification and Decryption**: This process decrypts the metadata and verifies the signature of metadata. Signature verification confirms that message has arrived from the authenticated source. Request is processed further only after successful decryption and verification of the sender signature.
- **Decryption**: Purpose of this process is to decrypt the incoming file. It gets private keys from the Key Manager.
- **Malware Check**: This process checks the file against malware eradicate framework, which includes checking file against anti-virus engine, checks if file extension matches actual file content and performs content disarm and reconstruction.
- **File Distribution**: Dispatches the file to the destination folder based on metadata
- **Message Distribution**: Dispatches the text message to the destination rabbitmq queue.
- **Audit Log**: The purpose of this process is to record the start and end entries of each process for every request. These logs can be audited to ensure traceability of individual requests.
- **Routing Process**: This process routes messages from one process to another.
- **Transfer Request Status Update**: Purpose of this process is to update the status of the transfer request.

## 5.2. Setup

This section explains various server setup, purpose of the servers and third party software/hardware used.

### 5.2.1. Application Server Setup

#### Air-gapped server configuration

The purpose of this application server is to host services of the sender component for the files going out of air-gapped network and also the services of the receiver component for the files coming to air-gapped network. Hardware and OS configuration of the server is as follows:

- Intel(R) Xeon(R) Gold 6326 CPU@2.90 GHz (2 Processors)
- 8 GB RAM
- Windows Server 2019 Datacenter, 64-bit Operating System

#### Connected network server configuration

This application server hosts services of receiver component for the files coming out of air-gapped network and also the services of the sender component for the files going in air-gapped network. Hardware and OS configuration of the server is as follows:

- Intel(R) Xeon(R) Gold 6326 CPU@2.90 GHz (2 Processors)
- 8 GB RAM
- Windows Server 2019 Datacenter, 64-bit Operating System

#### Malware Eradicate Server configuration

To establish the Malware Eradication Framework, a commercial product is procured from a third-party vendor. This product comprises two main components: the first is a service responsible for downloading package updates from the internet, and the second is a core component that performs file security verification. The update service must be installed on a server with internet access, while the core component responsible for security checks is deployed on the internal server. It is possible to configure multiple nodes for the core component. However, only one node is used in the implementation. Its hardware and software configuration is follows:

- Intel(R) Xeon(R) Gold 6326 CPU@2.90 GHz (2 Processors)
- 8 GB RAM
- Windows Server 2019 Datacenter, 64-bit Operating System

#### RabbitMQ Server configuration

RabbitMQ is an open source messaging broker to exchange the message between producer and consumer. It is possible to install RabbitMQ in the cluster mode. However, for the implementation only single node of the RabbitMQ is installed. Configuration of RabbitMQ Server is as follows:

- Intel(R) Xeon(R) Gold 6326 CPU@2.90 GHz
- 8 GB RAM
- Windows Server 2019 Datacenter, 64-bit Operating System

### 5.2.2. Third Party Software and Hardware

Here is the list of third party software/hardware used:

#### RabbitMQ

RabbitMQ is a reliable and mature messaging and streaming broker. It facilitates communication and data exchange between applications through messages. RabbitMQ receives messages from Producers, stores them in queue and delivers them asynchronously to consumers [26]. It is used to achieve asynchronous behavior in file transfer processes. How RabbitMQ helps in achieving asynchronous behavior and how its routing mechanism helps in processing of the request is shown in fig. 5.2.

The flow diagram illustrates RabbitMQ message exchange and Queue processing:

**Step 1:**

The process begins with the Transfer Initiation Process, which sends a message containing a payload and a routing key to the RabbitMQ Exchange.

**Step 2:**

The exchange distributes the message to two distinct queues:

- The Audit Start Queue, used for logging start of the process
- designated queue as defined by the channel configuration based on provided routing key, targeted for further processing.

**Step 3:**

Messages queued in Step 2 are retrieved and processed by their respective subscribers:

- The Audit Start process retrieves the message from Audit Start queue and marks the beginning of the process for request processing.

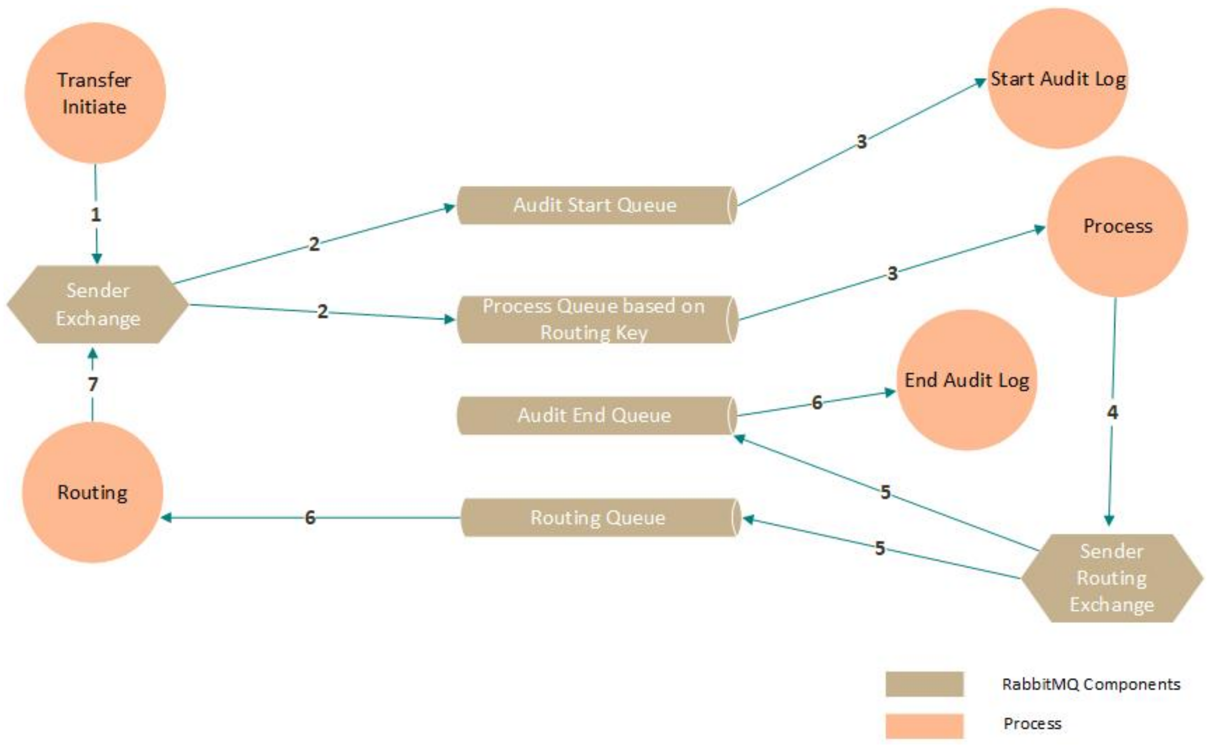

Figure 5.2.: RabbitMQ Message Exchange and Queue Processing Flow

- Another process (specified by channel configuration) concurrently retrieves and processes the message from its respective queue.

**Step 4:**

Upon successful processing by both subscribers, each processed message is forwarded to the Routing Exchange.

**Step 5:**

The Routing Exchange then directs the incoming message to two queues:

- Audit End Queue, marking the completion of audit tracking for the process.
- Routing Queue, managing further message routing within the channel pipeline.

**Step 6:**

- The Audit End Process retrieves the message from the Audit End Queue, completes logging, and concludes the audit cycle for that particular process of the request processing.
- Simultaneously, the Routing Process fetches the message from the Routing Queue, removes the first routing key from the channel pipeline, and resubmits the updated message to the Sender Exchange. This ensures the message continues through the pipeline and is subsequently handled by the next configured process.

### Data Diode

A Data diode is a commercial hardware product which is used to transfer the files between two security domains. Basically, it is composed of two 19" appliances. Each appliance is deployed on one domain. Both the appliances communicate with each other through passive device. It is administered from the high security domain. File transfers in data diode are through channels [27]. Channel comes with three different priorities: High, Medium and Low.

**Definition of Channel**: A channel represents a link between a source file location and a destination file location across two networks — one considered "external" and one "internal".

**How It Works:**

- **Source Access**:

  The external appliance acts as a client to a file server, using one of the standard file transfer protocols:
  - FTP
  - FTPS
  - SFTP
  - SMB
- **Transfer Over Diode**: Files are automatically transferred from the external appliance to the internal one — this is typically a one-way, hardware-enforced data flow.
- **Destination Upload**: The internal appliance then connects (again, as a client) to its respective file server and uploads the files to the predefined destination folder.

**Integration:** The system is designed to fit into existing infrastructure with no need for changes to how file servers are managed — it uses common, well-supported protocols.

### Malware Eradicate Framework

A commercial product is procured to ensure file security. The product consists of two primary components. The first is a service that handles the downloading of various package updates, including antivirus engine definitions, file type validation rules, and Content Disarm and Reconstruction (CDR) updates. The second is the core component, which is responsible for performing the actual file security checks. It can scan files using an antivirus engine and sanitize them using CDR technology [28].

The core component provides a REST API for submitting files for security verification. It also supports configurable workflows that define parameters such as whitelisted and blacklisted file types, maximum file size limits, and other validation criteria. These workflows can be configured on a per-channel basis, allowing the transferring application to determine which specific workflow to use for file security evaluation.

## 5.3. Implementation

The middleware application responsible for transferring files between networks is developed using the following technology stack:

- .NET Framework 4.8
- ASP.NET Web API
- Windows Service
- RabbitMQ 6
- Microsoft SQL Server 2019
- Internet Information Services (IIS)

The following outlines the overall process from the perspective of consumer application:

- The middleware exposes two REST API endpoints:
  - File Upload API: Accepts a file and returns a unique file identifier upon successful upload
  - Transfer Initiation API: Accepts the unique file identifier along with necessary metadata to initiate the file transfer. Upon success, it returns a unique request ID.
- The unique request ID returned from the second API can be used to track the request status via a dedicated status-check API
- The consumer application follows a two-step sequence:
  1. Upload the file using the first API method
  2. Initiate the file transfer using the second API method with the file ID and metadata
- All background processing is handled by Windows Services, which:
  - Continuously poll RabbitMQ queues for messages.
  - Process the messages according to business logic.
  - Forward processed messages to the routing exchange.
  - Ensure that messages are moved through the defined processing pipeline to the next stage.

### 5.3.1. Air-Gapped to Connected Network Workflow

This section describes the workflow for transferring files from an air-gapped network to a connected network. The file transfer request is handled on both sides—within the air-gapped network and the connected network—passing through a request processing pipeline. The process is divided into two stages: one handled by the sender component, located in the air-gapped network, and the other by the receiver component, located in the connected network. The following two subsections provide a detailed explanation of how each component processes the request.

### Sender Component Workflow

Request processing pipeline by the sender component in the air-gapped network is depicted in fig. 5.3. The request is processed as follows:

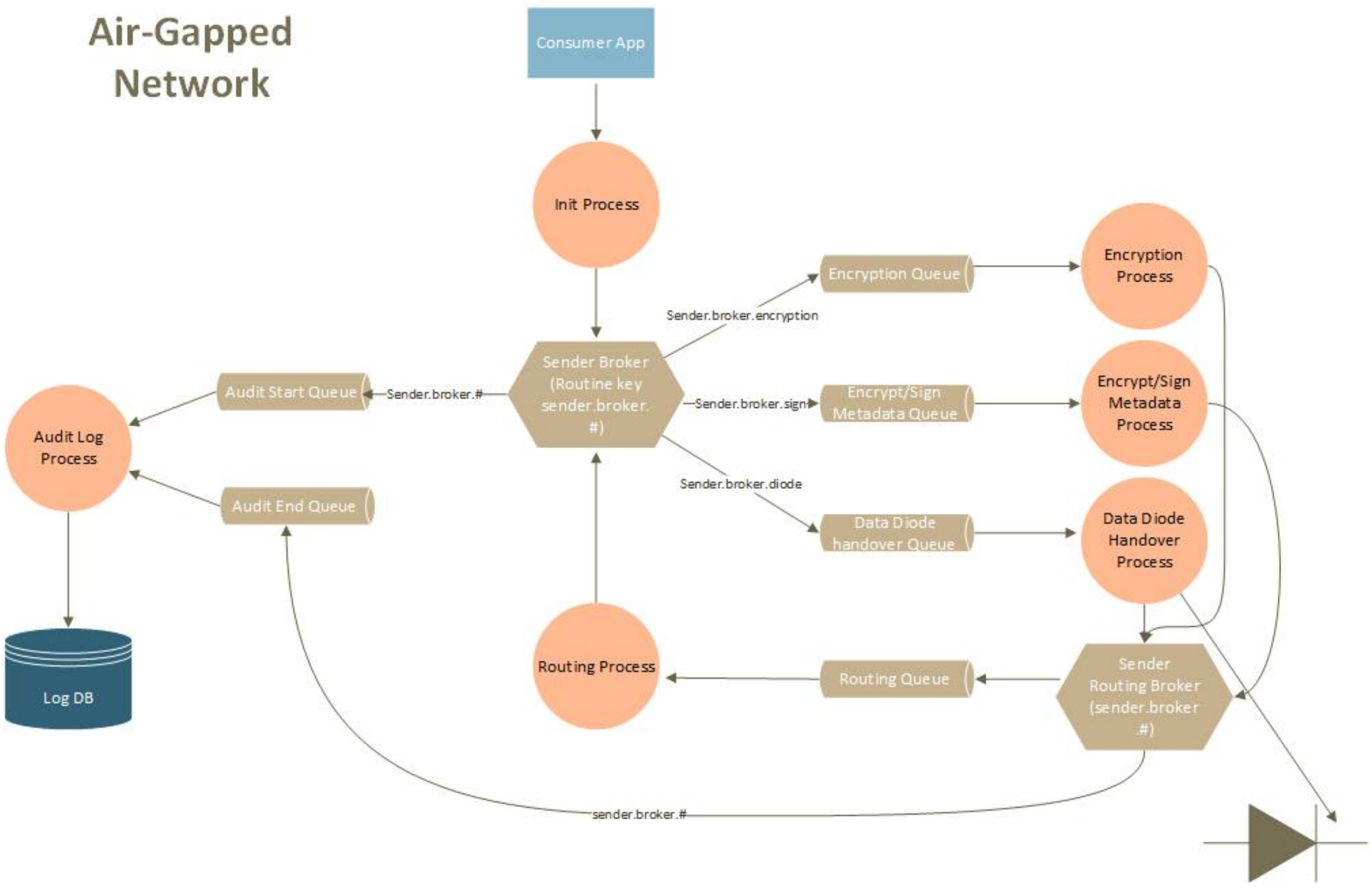

Figure 5.3.: Sender component worflow in air-gapped network

- The consumer app uploads the file (code snippet in listing 1) and initiates the file transfer process with metadata for the file transfer request. The Init process, along with channel configuration, sends the message to RabbitMQ exchange "Sender Broker". The channel configuration also contains a list of routing keys that determines the order in which the process is called while processing the transfer request.

```
using (HttpClient httpClient = new HttpClient())
{
    HttpResponseMessage response = await
     ↪  httpClient.PostAsync(endpointUrl, content);
    response.EnsureSuccessStatusCode();
}
```

Listing 1: File upload through Web API

- Sender Broker exchange sends all the messages that start with the routing key sender.broker to the Audit Start queue. Audit Log process retrieves the message and marks the beginning of the process. Each routing key starts with sender.broker, which means that Sender Exchange sends every message to the Audit Log process.
- The first routing key in the request pipeline is sender.broker.encryption. Sender exchange sends the message to Audit Start queue, to be logged by Audit Log process, and also to the Encryption queue. The Encryption process retrieves the message from the queue, encrypts the file, and sends the message to the Sender Routing Broker exchange. Code snippets to create rabbitmq connection, polling, receiving and processing of the message, and acknowledging the message is shown in listing 2, listing 3, listing 4 and listing 5 respectively.

```
var factory = new ConnectionFactory()
{
    HostName = "localhost",   // RabbitMQ host name
    UserName = "guest",       // username
    Password = "guest"        // password
    VirtualHost = "virtualHost" //Virtual Host Name
};

using (var connection = factory.CreateConnection())
```

Listing 2: RabbitMQ Connection

- Sender Routing exchange sends all the messages that start with sender.broker to the Audit End queue.

Audit Log process retrieves the message and marks the completion of the process. In this case, the Audit process marks the completion of the Encryption process. Sender Routing Broker exchange also sends all the messages to Routing queue. The routing process gets the message, removes the first routing key from the request processing pipeline as it is processed, and sends the message to the Sender Broker exchange. Code snippet to publish message to rabbitmq exchange is shown in listing 6.

- The next routing key in the pipeline is sender.broker.sign. Exchange sends the message to the Encrypt/Sign Metadata queue which is processed by the Encrypt/Sign Metadata process, and the beginning of the process is also marked by the Audit Log process.

```
var channel = connection.CreateModel();
string queueName = "rabbitmqqueue"; // RabbitMQ queue name

// Ensure the queue exists before trying to consume
channel.QueueDeclare(queue: queueName,
                     durable: true,
                     exclusive: false,
                     autoDelete: false,
                     arguments: null);

var consumer = new EventingBasicConsumer(channel);
```

Listing 3: Subscribe to RabbitMQ queue

- The message is sent to routing exchange, the end of the process is marked, and routing process sends the message to Sender Broker exchange with new routing key as sender.broker.diode.
- This message is sent to Data Diode Handover Queue which is eventually processed by the Data Diode Handover process. The beginning and end of the process is recorded by Audit process and file is handed over to data diode to be transferred to other network.

### Receiver Component Workflow

Request processing pipeline by the receiver component in the connected network is shown in fig. 5.4. The request is processed as follows:

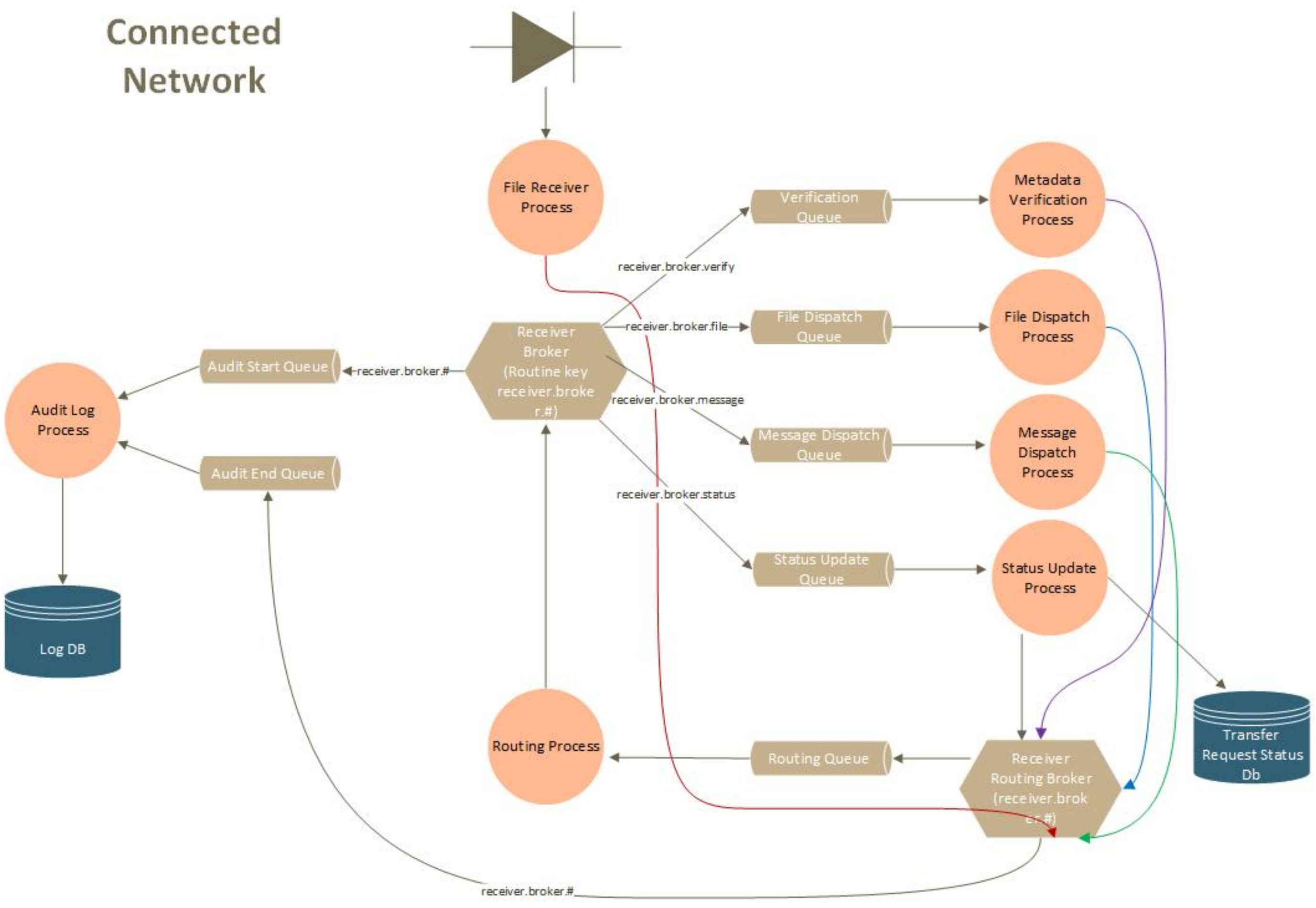

Figure 5.4.: Receiver component worflow in connected network

```
consumer.Received += (model, ea) =>
               {
                   var body = ea.Body.ToArray();
                   var message = Encoding.UTF8.GetString(body);

                   //Business Logic goes here
               };
// Start consuming
channel.BasicConsume(queue: queueName, autoAck: false, consumer:
 ↪   consumer);
```

Listing 4: Receive and Process RabbitMQ message

- On the receiver side, the workflow for recording the beginning and end of the process is the same as the sender component. Please not start of the file receiver process is not recorded due to the design how file receiver process is trigerred.
- The file receiver process receives the file from the Data Diode and sends the message to the receiver broker routing exchange. Routing broker forwards the message to the audit log process, which marks the completion of file receiver process, and to routing process that sends the message to receiver broker exchange for the further processing by the pipeline.

```
    //Acknowledge the message so it is removed from the queue
    channel.BasicAck(deliveryTag: ea.DeliveryTag, multiple: false);
```

Listing 5: Message Acknowledgment

- The receiver broker sends message to verification queue based on next configured routing key which is receiver.broker.verify. Metadata verification process retrieves the message, decrypts the metadata and verifies the signature to authenticate the sender of the message. Upon successful decryption and verification message is sent to routing exchange to mark the end of the process and to forward the message to broker again through routing process.
- Next, the receiver broker sends the message to file dispatch queue based on routing key receiver.broker.file. The file dispatch process retrieves the message and dispatches the file to the destination folder.
- Next, the message is sent to Message Dispatch queue. Message dispatch process retrieves the message from that queue and sends any text data to destination RabbitMQ queue.
- At last, after all the operations are successfully executed, file and messsage is delivered to destination, the message is processed by status update process which updates the status of the transfer request in the status update Db at the receiver side.

### 5.3.2. Connected to Air-Gapped Network Workflow

#### Sender Component Workflow

Request processing pipeline by the sender component in the connected network is shown in fig. 5.5. The request is processed as follows:

- Workflow for the sender component processes, for file transfer from connected network to air-gapped network, is identical to the workflow for sender component processes for file transfer from air-gapped

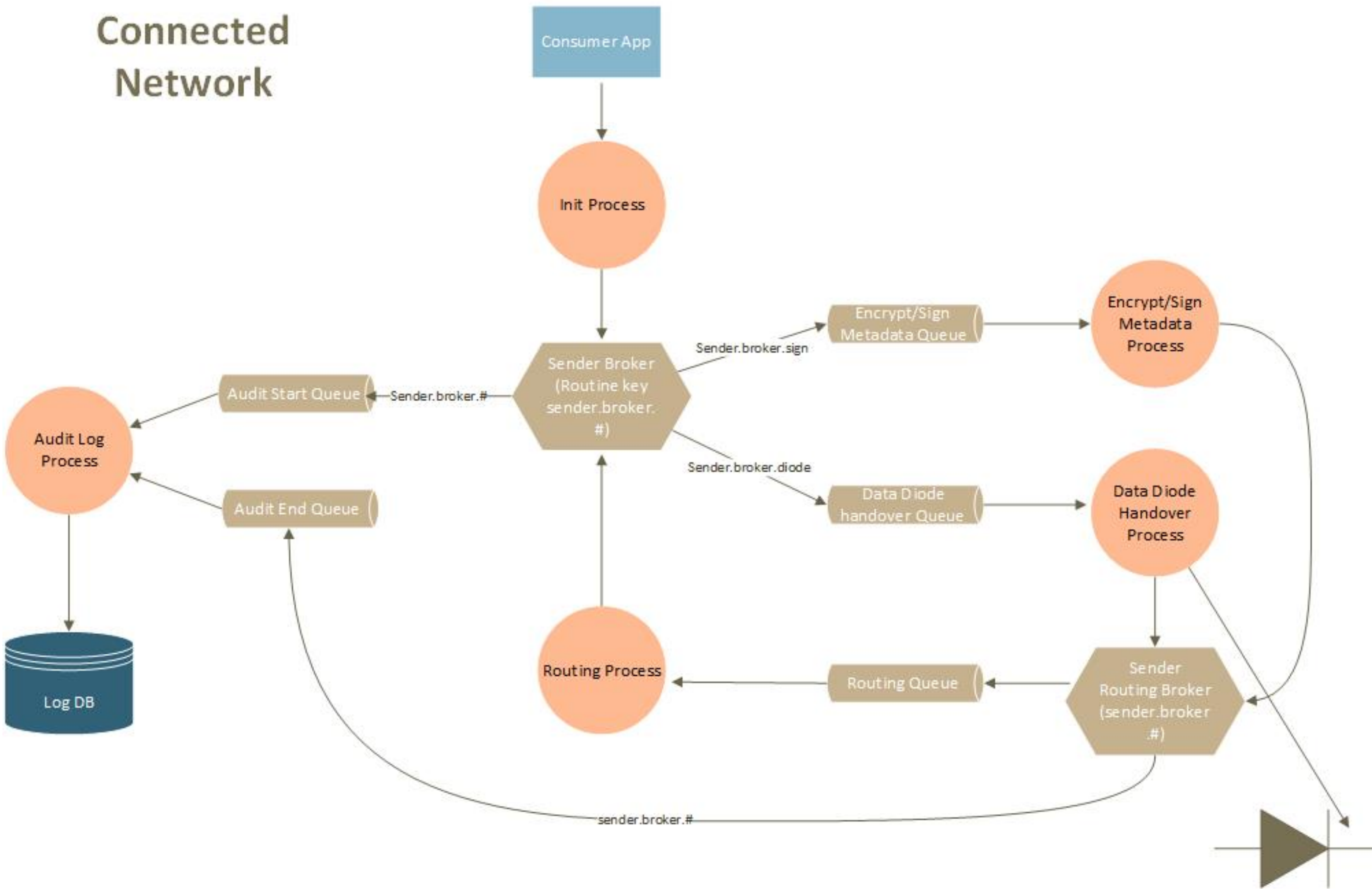

Figure 5.5.: Sender component worflow in connected network

network to connected network.

- The only exception to the workflow is absense of encryption process. As file moves from low security network and high security network, there is no need to encrypt the file. However, configuration of process is per channel and it is loosely coupled. If required for channel to encrypt the file when file is being transferred to air-gapped network, it is possible to achieve.

### Receiver Component Workflow

Request processing pipeline by the receiver component in the air-gapped network is depicted in fig. 5.6. The request is processed as follows:

- The receiver component flow on the air-gapped network side is same as receiver component flow on connected network side as described in subsection 5.3.1. However, there are some additional processes involved in the request processing pipeline.

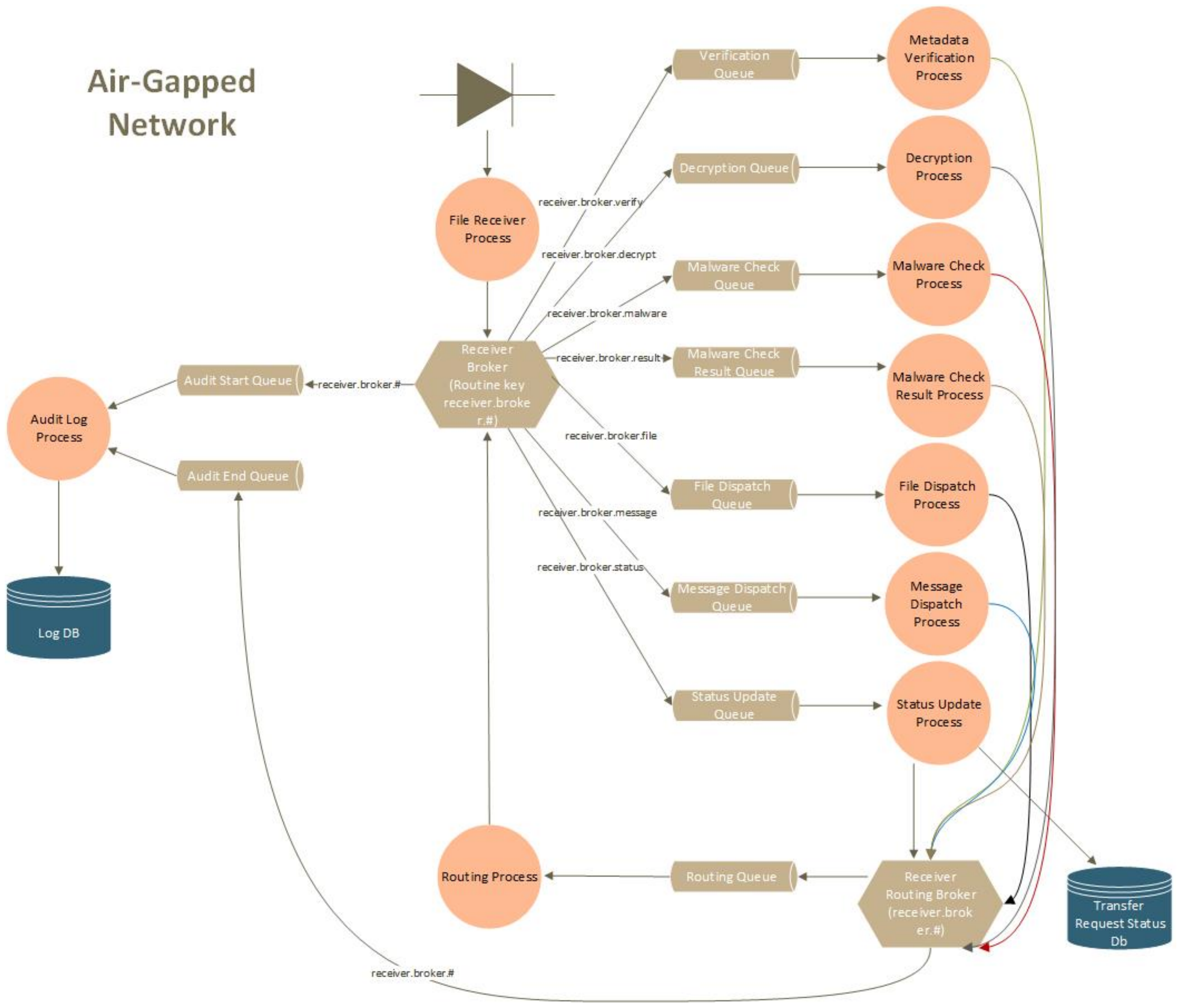

Figure 5.6.: Receiver component worflow in air-gapped network

```
// Declare the exchange if not exists
channel.ExchangeDeclare(exchange: "exchangeName",
                        type: "topic",
                        durable: true);

// Convert the message to a byte array
var body = Encoding.UTF8.GetBytes(message);

// Publish the message to the exchange with a routing key
channel.BasicPublish(exchange: "exchangeName",
                     routingKey: "routingKey",
                     basicProperties: null,
                     body: body);
```

Listing 6: Publish Message

- The first additional process is decryption process. It is possible that data is received from some external entity and it is encrypted for the entity available only in air-gapped network. In such scenario, decryption process decrypts the file.
- After decryption process comes the malware check process. A file to be checked against malware is submitted to malware eradicate tool through REST API. Based on file type, file size and total load on tool it may take some time.
- The process malware check result gets the result of malware check. If file has passed all the checks then next process involved is called and file is delivered to destination. If file does not pass all the checks, then file is quarantined and it goes to security team for further manual investigation.

### 5.3.3. File Transfer Acknowledgment Workflow

The overall workflow of file transfer acknowledgment is shown in fig. 5.7

- As described in fig. 5.4 and fig. 5.6, receiver component processes update the status of file transfer request. Status of the file transfer request, initiated in air-gapped network is updated in connected network and status of the file transfer request, initiated in connected network is updated in air-gapped network. The goal is to sync the status on the initiator side.
- The acknowledge process retrieves the file transfer status for the other side of network in the form

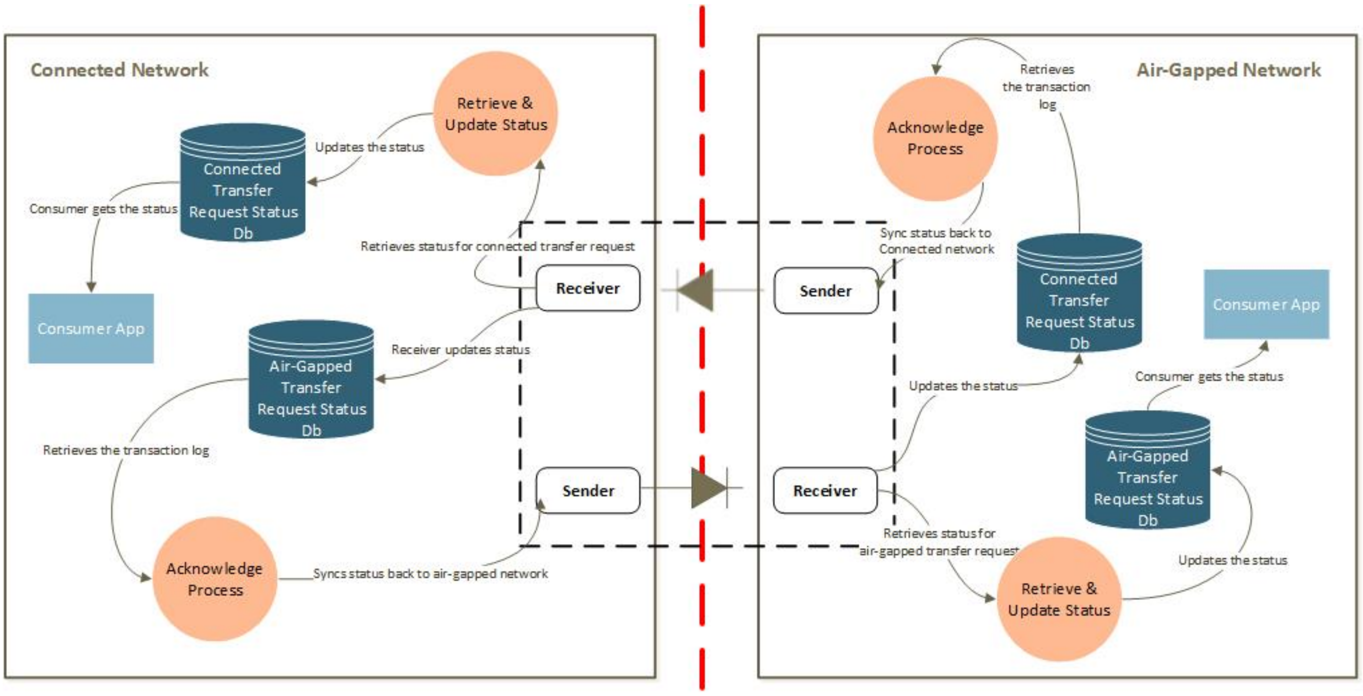
Connected Network
Retrieve & Update Status
Updates the status
Connected Transfer Request Status Db
Consumer gets the status
Consumer App
Retrieves status for connected transfer request
Receiver
Air-Gapped Transfer Request Status Db
Receiver updates status
Retrieves the transaction log
Sender
Acknowledge Process
Syncs status back to air-gapped network
Air-Gapped Network
Retrieves the transaction log
Acknowledge Process
Sync status back to Connected network
Sender
Connected Transfer Request Status Db
Consumer App
Updates the status
Consumer gets the status
Air-Gapped Transfer Request Status Db
Receiver
Retrieves status for air-gapped transfer request
Retrieve & Update Status
Updates the status

Figure 5.7.: File transfer acknowledgment workflow

of transaction log. This transaction log is transferred back to its originating side through middleware application. The update status process retrieves the status and updates it in originating database. Consumer app can get the message from the database.

- For example, air-gapped transfer request status is updated in "Air-Gapped Transfer Request Status" Db on connected network. At regular intervals (2 minutes) the acknowledge process gets the updated status of file transfer request from air-gapped network in the form of transaction log. These transaction logs are transferred back to the air-gapped network. On the air-gapped network, the update status process retrieves the status of transfer request from the receiver component and updates it back to "Air-Gapped Transfer Request Status" Db. Consumer App retrieves the status from this Db through REST API.

## 5.4. Limitations

This section lists the limitation of the middleware implementation used to transfer the files from one network to another network.

### 5.4.1. RabbitMQ Dependency

RabbitMQ is the central component in achieving asynchronous processing of file transfer request. As rabbitmq messages are stored on the disk and do not remain in the memory, messages survive rabbitmq server reboot. However, With single node installation of rabbitmq, if rabbitmq crashes, it can completely stop the file transfer. Thus, rabbitmq remains the single point of failure.

### 5.4.2. Request Processing Order

Currently file transfer requests are processed on "first come first served" basis. RabbitMQ also maintains the queue in the order of the message arrival. Message that arrived first is sent to process first for the business logic processing. In case there is a long queue in the request processing pipeline, higher priority request has to wait until all the requests that arrived before have been processed.

### 5.4.3. File Size Limitation

Data Diode has a limitation of maximum supported file size of 20 GB [27]. File more than 20 GB in size cannot be transferred through Data Diode. Therefore, this implementation is not suitable for single file, more than 20 GB. However, large size file can be split into multiple smaller size files and transferred. Once all the files are received on the destination, they can be merged to create single large file.

### 5.4.4. Performance Bottlenecks

During testing, it was observed that the file transfer process takes longer when transferring from the connected network to the air-gapped network, compared to the reverse direction. The reason is additional step of scanning file against malware eradicate framework. Time taken by the malware eradicate framework to process the file depends on the type and size of the file. The larger the file the more time required to process the file. Same way the complex the file type, it requires more time to process the file, i.e. it takes more time to process the Microsoft Word Document compare to text files.

It is noticed that when 100 concurrent requests are made to transfer the Word documents of the size 1 MB, only 91 documents were processed successfully by malware eradicate framework. It indicates that when more concurrent request for the complex and large size file is normal, malware eradicate framework must be scaled out.

### 5.4.5. Unable to log start of the File Receiver Process

Audit process logs start and end of every processes. Request processing workflow is designed such a way that it routes to every process through broker exchange and broker routing exchange. Both exchanges trigger Audit process which logs start and end of the process. However, the file receiver process on the receiver side is not trigerred from any of the exchanges, so audit process is not called and start of the file receiver process is not logged. The next process after the file receiver process is called through routing exchange which causes end of the file receiver process to be logged.

# 6. Results

This section presents a series of test cases designed and executed to comprehensively evaluate the stability, performance, security, reliability, and scalability of the middleware application. Test results for the executed test cases are described in the following sub-sections:

## 6.1. Stability Testing

A reusable health-check component was developed to test the stability of the middleware application. A component is reusable in a way that same component can be used to check overall health check for file transfer in both the directions: a file transfer from air-gapped network to connected network and a file transfer from connected network to air-gapped network. A health check component comprises of two parts: a health check service and a health check web page. A detail description of both the parts are given below:

### 6.1.1. Health check service

The purpose of health check service is to make file transfer request at regular interval. The interval is configurable. This service was created using .net framework 4.8 and Windows Service project template in the Visual Studio IDE. This service is hosted along with sender component processes.

### 6.1.2. Health check web page

The purpose of this web page is to display status of overall health check of both sender and receiver components. File transfer request is successfully delivered only when all the processes of sender and receiver components are running. Health check web page checks file destination and RabbitMQ queue destination. If both file and text message is delivered to destination, health check is considered to be passed otherwise failed. A health check web page displays "True" is health check testis passed or "False" if health check test has failed.

| Intervals (Hours) | Status | Success Rate |
|---|---|---|
| 2 | Pass | 100% |
| 4 | Pass | 100% |
| 6 | Pass | 100% |
| 8 | Pass | 100% |
| 23 | Pass | 100% |
| 27 | Pass | 100% |
| 31 | Pass | 100% |
| 46 | Pass | 100% |
| 54 | Pass | 100% |
| 69 | Pass | 100% |

Table 6.1.: Health check test results

A health check web page is monitored manually at regular intervals for the first 72 hours. Its result is shown in table 6.1 which indicates that tests were successful:

## 6.2. Performance Testing

Performance testing was carried out with different types and sizes of files. Concurrent file transfer requests were made and total time taken by all the files to reach the destination was noted. JMeter is used to initiate the file transfer request. JMeter has the feature to initiate the concurrent requests, which is used to generate concurrent request for different files.

### 6.2.1. Test result for TEXT files

Performance test result for the text file types of different sizes and concurrent requests, transferred from air-gapped network to connected network is shown in table 6.2, and for the files transferred from connected network to air-gapped network is shown table 6.3.
From the table comparison, it is clearly visible that files transferred from connected network to air-gapped network take slightly higher time compared to files transferred from air-gapped network to connected network. One main reason for this difference is higher number of processes in request processing pipeline for the files coming to air-gapped network.

| File Size | Concurrent Requests | Total Time (in sec) | Success Rate |
|---|---|---|---|
| | 10 | 29 | 100% |
| 10 KB | 50 | 41 | 100% |
| | 100 | 60 | 100% |
| | 10 | 42 | 100% |
| 100 KB | 50 | 55 | 100% |
| | 100 | 71 | 100% |
| | 10 | 21 | 100% |
| 1 MB | 50 | 52 | 100% |
| | 100 | 65 | 100% |

Table 6.2.: Air-Gapped initiated Text Files Performance Test Result

| File Size | Concurrent Requests | Total Time (in sec) | Success Rate |
|---|---|---|---|
| | 10 | 57 | 100% |
| 10 KB | 50 | 70 | 100% |
| | 100 | 110 | 100% |
| | 10 | 50 | 100% |
| 100 KB | 50 | 66 | 100% |
| | 100 | 109 | 100% |
| | 10 | 48 | 100% |
| 1 MB | 50 | 70 | 100% |
| | 100 | 139 | 100% |

Table 6.3.: Connected network initiated Text Files Performance Test Result

### 6.2.2. Test result for Word Documents

table 6.4 shows the performance test result for the Microsoft Word documents (.docx files), transferred from air-gapped network to connected network. While table 6.5 shows the performance test result for .docx files transferred from connected network to air-gapped network.

| File Size | Concurrent Requests | Total Time (in sec) | Success Rate |
|---|---|---|---|
| 13 KB | 10 | 17 | 100% |
| | 50 | 49 | 100% |
| | 100 | 65 | 100% |
| 100 KB | 10 | 19 | 100% |
| | 50 | 47 | 100% |
| | 100 | 66 | 100% |
| 1 MB | 10 | 34 | 100% |
| | 50 | 52 | 100% |
| | 100 | 81 | 100% |

Table 6.4.: Air-Gapped initiated Docx Files Performance Test Result

| File Size | Concurrent Requests | Total Time (in sec) | Success Rate |
|---|---|---|---|
| 13 KB | 10 | 70 | 100% |
| | 50 | 254 | 100% |
| | 100 | 409 | 100% |
| 100 KB | 10 | 81 | 100% |
| | 50 | 236 | 100% |
| | 100 | 636 | 100% |
| 1 MB | 10 | 144 | 100% |
| | 50 | 632 | 100% |
| | 100 | 1110 | 91% |

Table 6.5.: Connected network initiated Docx Files Performance Test Result

From the comparison of the tables, it is clearly visible that word documents transferred from connected network to air-gapped network take considerably higher time compare to those transferred from air-gapped network to connected network. As the size the file and concurrent requests increase, the difference is exponential. The reason for this delay is time taken by malware eradicate process to process the word file. As the word document is just not single file but it is collection of multiple files, it requires more time in processing.

Another reason is malware check result process. When malware check result process attempts to retrieve result from the malware processing and at the same file is still being processed, check result process makes another attempt. Time interval between every attempts gradually increase which is also the cause for the delay.

### 6.2.3. Test result for PDF files

Performance test result for pdf file types, transferred from air-gapped network to connected network is shown in table 6.6, and for the files transferred from connected network to air-gapped network is shown in table 6.7.
The result shows that pdf files transferred from connected network to air-gapped network takes more time. Again, the reason is more number of processed in the request processing pipeline in air-gapped network and also time taken by malware check process to process the pdf files. As it is evident that as size of the file increases, time taken to deliver the file doubled up.

| File Size | Concurrent Requests | Total Time (in sec) | Success Rate |
|---|---|---|---|
| 246 KB | 10 | 16 | 100% |
| | 50 | 45 | 100% |
| | 100 | 67 | 100% |
| 685 KB | 10 | 21 | 100% |
| | 50 | 49 | 100% |
| | 100 | 86 | 100% |
| 1.3 MB | 10 | 25 | 100% |
| | 50 | 66 | 100% |
| | 100 | 82 | 100% |

Table 6.6.: Air-Gapped initiated PDF Files Performance Test Result

## 6.3. Security Testing

Security testing involves verifying data integrity and ensuring confidentiality. Integrity and confidentiality checks apply at two levels: data level and metadata level. Checks at both the level are explained in detail in following sub-sections:

| File Size | Concurrent Requests | Total Time (in sec) | Success Rate |
|---|---|---|---|
| 246 KB | 10 | 39 | 100% |
| | 50 | 89 | 100% |
| | 100 | 137 | 100% |
| 685 KB | 10 | 52 | 100% |
| | 50 | 137 | 100% |
| | 100 | 278 | 100% |
| 1.3 MB | 10 | 154 | 100% |
| | 50 | 300 | 100% |
| | 100 | 520 | 100% |

Table 6.7.: Connected network initiated PDF Files Performance Test Result

### 6.3.1. Metadata level

Before a file transfer request crosses the network boundary, its metadata is encrypted using the target service account's public key and digitally signed using the private key of the sending process's service account. If metadata cannot be encrypted or signed at the sender side, file transfer request processing stops immediately. Similarly, if metadata cannot be decrypted or signature cannot be verified at the receiver side, the file transfer request does not proceed further.

The above use case is validated by removing one of the following keys one at a time:

- Private key of sending service account
- Public key for receiving service account
- Decryption key for receiving service account
- Verification key for sending service account

It is successfully tested that if any of the above keys is missing, file transfer request processing halts.

### 6.3.2. Data level

File is encrypted for the target user when it is being transferred from air-gapped network to connected network. As the sensitive data moves from highly secure network to less secure network, preserving data confidentiality is very much needed. However, when file is being transferred from connected network to air-gapped network, it is not encrypted. When file going from air-gapped network to connected network

cannot be encrypted for the target user, file transfer request processing stops immediately.
The above use case is tested by removing public key for the target user from the air-gapped network. It is tested successfully that when public key for the target user is not available in the air-gapped network, file cannot be encrypted and it cannot be transferred to connected network. Encrypted file can only be decrypted using user's decryption key which is available only to target user in security token.

## 6.4. Reliability Testing

The purpose of reliability testing is to verify that application is fault tolerant. In the context of current application, reliability testing involves making sure that file is either delivered successfully or remains in the processing pipeline irrespective of any fault. Reliability testing was carried out by eliminating connections between dependent systems. Following services were stopped and its impact on the application was assessed:

### 6.4.1. RabbitMQ

When RabbitMQ service is stopped, none of the processes receive the message from the RabbitMQ, however, messages still remain in the RabbitMQ queues. Files are not delivered to destination till the time RabbitMQ service remains stopped. As soon as the service is started, request processing pipeline starts receiving messages from the queues, processes the messages, and files are delivered to destination.

### 6.4.2. SQL Server

When database service SQL Server is stopped, audit log process is stopped due to the error with database connection. However, it does not impact request pipeline processing. File transfer request are still being processed and files are sent to destination.
As audit log process is stopped, process start and end entries are not stored in database. Until the audit logs are stored in database, status service returns incorrect status for the file transfer request.

### 6.4.3. Malware Eradicate Service

When malware eradicate service is stopped, it does not impact file request processing cycle for the files being transferred from air-gapped network to connected. However, file transfer request from connected network to air-gapped network is impacted. Messages remain in the error queue which need to be moved back to malware check queue to be processed again.

## 6.5. Scalability Testing

The purpose of scalability testing is to ensure that the solution can scale effectively when increased processing power is needed, particularly for concurrent large-scale processing. The solution's components can be categorized into two types: in-house developed components and third-party components, such as RabbitMQ and the malware eradication product. According to the documentation provided by the third-party vendors, both RabbitMQ and the malware eradication tool support scalability and allow for multi-node installations. To enable a scale-out scenario for the in-house component, multiple instances of the request processing service—responsible for consuming messages from RabbitMQ—were deployed across several servers. Each service instance can independently subscribe to the same RabbitMQ queue and retrieve messages. This horizontal scaling approach increases the overall processing capacity.

# 7. Discussion

The previous chapters have demonstrated methods for exchanging data securely and in a controlled manner between two networks operating under different security domains. They detailed strategies for maintaining confidentiality during the transfer of data from a high-security network to a lower-security network, as well as techniques for protecting the high-security network from incoming traffic.

In this chapter, we further discuss approaches for enhancing the performance of high-priority file transfers and examine strategies for preserving data confidentiality within the lower-security network. Additionally, potential improvements and limitations of the proposed methods are analyzed, providing insight into future developments in secure inter-network communication.

## 7.1. Information Lifecycle Management Enforcement

Confidential and sensitive data transferred from air-gapped network to connected network is encrypted for the target users within an organization or external users. It is possible when data is encrypted for internal users, users can decrypt the data and keep data in decrypted form for longer or forever.

Part of an information lifecycle management system is the creation of a data policy that categorizes types of data and establishes whether a particular data is allowed to have it on a particular device. Then a data discovery tool can be employed to help enforce the policy. At the very least the discovery tool finds files by name and type and often will scan the files for the existence of critical data.

To implement a data sanitization program at the file level, the organization has to decide what types of data can be stored on what types of devices. After content is discovered on a device that violates the data policy, a command can be sent over the network to sanitize it. Sanitizing a file requires overwriting. Each sector that contains segments of the file is overwritten with ones and zeroes pursuant to the desired level of insurance against forensic analysis. A record should be kept of the erasure, preferably in a central location. The records should be tamperproof and auditable [23] [29].

## 7.2. Data Diode Channel Priority

Currently, all file transfer requests are handled with equal priority. However, the data diode allows for setting different channel priorities: High, Medium, and Low. It is recommended to configure at least a couple of data diode channels with high priority. When a high-priority request is received, it should be routed through a high-priority data diode channel. Since RabbitMQ processes all requests equally on a first-come, first-served basis, using the data diode's priority channels enables prioritization at the receiving side. Although all requests are initially treated with the same priority at the sender side, the data diode ensures that high-priority requests are transferred to the destination network ahead of others.

## 7.3. Retry Mechanism

Every process of request processing pipeline returns one of the three statuses: success, failure and retry. In case of success status, next process in the request processing pipeline is called. When process returns failure status, request processing flow stops immediately. When process returns retry status, process re-attempts the business logic. The number of re-attempts and frequency can be configured per process. As lots of processing can be involve when more number of re-attempts are configured, process must return retry status carefully and also number of re-attempts and it frequency must carefully configured.

## 7.4. Data Loss Prevention (DLP)

It is possible to transfer arbitrary files from air-gapped network to connected network through approval-based channel. Even though, there is a possibility by a approver to review the file being transferred, it is still possible to hide something and transfer the sensitive content without being noticed. To avoid such scenario, Data Loss Prevention (DLP) should be implemented. DLP can be of different types like Endpoint DLP, Network DLP, Storage DLP, Cloud DLP and Application DLP [30]. Essential capabilities of effective DLP are Manage, Discover, Monitor, and Protect [31].

# 8. Conclusion

This thesis introduced a multi-layered architecture that allows secure data transfer through a Data Diode while maintaining strict control over the process. The proposed solution ensures authorized users can transfer pre-approved data types only. The capability logging the beginning and end of each process for every file transfer request guarantees accountability and attribution of the file transfer request. The proposed workflow ensures protection of air-gapped network from malicious incoming file. The proposed design presents a solution to overcome the limitation of one-way communication by using an alternative approach to acknowledge file transfer requests. This research demonstrates a method for securely and efficiently transmitting essential data across networks that use enforced one-way data flow.
The findings show great potential yet solid groundwork but research always remains an unfinished process. Ongoing research needs continuous examination and enhancement to address emerging problems. Following sections present various potential paths for upcoming research endeavors.

## 8.1. Suggestions for Future Work

### 8.1.1. RabbitMQ Clustering

As per current implementation, there is a single node installation of RabbitMQ. Single node installation could potentially cause the single point of failure. To overcome the risk associated with single node installation, more than one RabbitMQ nodes should be installed to form RabbitMQ cluster [32]. A RabbitMQ cluster is a logical grouping of one or more (three, five, seven, or more) nodes, each sharing users, virtual hosts, queues, streams, exchanges, bindings, runtime parameters and other distributed state. In the cluster setup, RabbitMQ brokers tolerate the failure of individual node.

### 8.1.2. Request processing based on priority

The middleware application processes the file transfer request based on the order it is received. Consider a scenario where there is a long queue of requests to be processed, and then comes the high priority request which needs to be processed on the high priority basis. All the request processing pipeline processes retrieves the messages, to be processed, from the rabbitmq queue. RabbitMQ queue also returns the message based on the order it received the messages. However, RabbitMQ queue has a feature called "Priority Queue" [33]. Priority values between 1 and 255 are supported. Higher priority values require more CPU and memory resources. Therefore, it is highly recommended to use priority values between 1 and 5. Priority queue feature can be leveraged by the processes to process the file transfer request with higher priority first based on assigned priority to request.

# List of Figures

# List of Tables

# List of Listings

# Acronyms

| | |
|---|---|
| ACL | Access Control List |
| AES | Advanced Encryption Standard |
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| APT | Advanced Persistent Threat |
| ASCII | American Standard Code for Information Interchange |
| CD | Compact Disc |
| CDP | Cross Domain Problem |
| CDR | Content Disarm and Reconstruction |
| CDS | Cross Domain Solution |
| CPU | Control Processing Unit |
| CTF | Capture The Flag |
| DDoS | Distributed Denial of Service |
| DES | Data Encryption Standard |
| DLP | Data Loss Prevention |
| FE | Functional Encryption |
| FTP | File Transfer Protocol |
| FTPS | File Transfer Protocol Secure |
| GB | Gigabyte |

| | |
|---|---|
| HE | Homomorphic Encryption |
| HIPAA | Health Insurance Portability and Accountability Act |
| ICS | Industrial Control System |
| IDE | Integrated Development Environment |
| IDS | Intrusion Detection System |
| IIS | Internet Information Services |
| IoT | Internet of Things |
| IPS | Intrusion Prevention System |
| ISO | International Organization for Standardization |
| IT | Information Technology |
| JSON | JavaScript Object Notation |
| LAN | Local Area Network |
| Malware | Malicious computer software |
| OS | Operating System |
| OSS | Open Source Software |
| PII | Personally Identifiable Information |
| RAM | Random Access Memory |
| Ransomware | Malware, preventing user from accessing system or files, demanding ransom money |
| REST | Representational State Transfer |
| SFTP | SSH File Transfer Protocol |
| SMB | Server Message Block |

| | |
|---|---|
| TCP/IP | Transmission Control Protocol/Internet Protocol |
| USB | Universal Service Bus |
| VLAN | Virtual Local Area Network |
| VPN | Virtual Private Network |

# Bibliography

[1] Jangyong Park, Jaehoon Yoo, Jaehyun Yu, Jiho Lee, and JaeSeung Song, "A survey on air-gap attacks: Fundamentals, transport means, attack scenarios and challenges," *Sensors*, vol. 23, no. 6, 2023, ISSN: 1424-8220. DOI: 10.3390/s23063215. [Online]. Available: https://www.mdpi.com/1424-8220/23/6/3215.

[2] "Microsoft digital defense report 2024," *Intelligence Report*, [Online]. Available: https://www.microsoft.com/en-us/security/security-insider/intelligence-reports/microsoft-digital-defense-report-2024#section-master-oc526b.

[3] "Cybersecurity forecast 2025," *Google Cloud Security*, [Online]. Available: https://www.gstatic.com/gumdrop/files/cybersecurity-forecast-2025.pdf.

[4] Mohan Rajkumar Na and Sundharakumar K B, "A study on air-gap networks," in *2024 5th International Conference on Innovative Trends in Information Technology (ICITIIT)*, 2024, pp. 1–6. DOI: 10.1109/ICITIIT61487.2024.10580382.

[5] Mordechai Guri, *Mind the gap: Can air-gaps keep your private data secure?* 2024. arXiv: 2409.04190 [cs.CR]. [Online]. Available: https://arxiv.org/abs/2409.04190.

[6] V. Dhanush, A. R Mahendra, M V Kumudavalli, and Debabrata Samanta, "Application of deep learning technique for automatic data exchange with air-gapped systems and its security concerns," in *2017 International Conference on Computing Methodologies and Communication (ICCMC)*, 2017, pp. 324–328. DOI: 10.1109/ICCMC.2017.8282701.

[7] Silverfort, "Air gap network," [Online]. Available: https://www.silverfort.com/glossary/an-air-gapped-network/.

[8] Mohammad Tazeem Naz and Ahmed M. Zeki, "A review of various attack methods on air-gapped systems," in *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*, 2020, pp. 1–6. DOI: 10.1109/3ICT51146.2020.9311995.

[9] Ralph Langner, "Stuxnet: Dissecting a cyberwarfare weapon," *IEEE Security & Privacy*, vol. 9, no. 3, pp. 49–51, 2011. DOI: 10.1109/MSP.2011.67.

[10] Fatemeh Khoda Parast, "Equation unsolved: Inside the shadowy world of elite cyber spies," in *2024 34th International Conference on Collaborative Advances in Software and COmputiNg (CASCON)*, 2024, pp. 1–6. DOI: 10.1109/CASCON62161.2024.10838060.

[11] Reyner Aranta Lika, Danushyaa Murugiah, Sarfraz Nawaz Brohi, and Daksha Ramasamy, "Notpetya: Cyber attack prevention through awareness via gamification," in *2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*, 2018, pp. 1–6. DOI: 10.1109/ICSCEE.2018.8538431.

[12] Annegret Bendiek and Matthias Schulze, "Attribution: A major challenge for eu cyber sanctions. an analysis of wannacry, notpetya, cloud hopper, bundestag hack and the attack on the opcw," eng, Berlin, SWP Research Paper 11/2021, 2021. DOI: 10.18449/2021RP11. [Online]. Available: https://hdl.handle.net/10419/253242.

[13] Vassilii Arkhangelskii, Anna Epishkina, Vadim Kalmykov, and Konstantin Kogos, "Secure one-way data transfer," in *2016 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW)*, 2016, pp. 392–395. DOI: 10.1109/EIConRusNW.2016.7448203.

[14] Ismail Ahmed Almaazmi, Mohammed Saeed Al Shehhi, Omar Ahmed Alkhoori, Salem Jumah Al Shehhi, and Yasir Hamid, "Data diode for cyber-security: A review," in *2022 International Conference on Artificial Intelligence of Things (ICAIoT)*, 2022, pp. 1–6. DOI: 10.1109/ICAIoT57170.2022.10121887.

[15] Masike Malatji, "Application of data diodes in internet of things security," in *2021 International Conference on Electrical, Computer and Energy Technologies (ICECET)*, 2021, pp. 1–4. DOI: 10.1109/ICECET52533.2021.9698482.

[16] Hamed Okhravi and Fredrick T. Sheldon, "Data diodes in support of trustworthy cyber infrastructure," in *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, ser. CSIIRW '10, Oak Ridge, Tennessee, USA: Association for Computing Machinery, 2010, ISBN: 9781450300179. DOI: 10.1145/1852666.1852692. [Online]. Available: https://doi.org/10.1145/1852666.1852692.

[17] R.T. Barker and C.J. Cheese, "The application of data diodes for securely connecting nuclear power plant safety systems to the corporate it network," in *7th IET International Conference on System*

*Safety, incorporating the Cyber Security Conference 2012*, 2012, pp. 1–6. DOI: 10.1049/cp.2012.1514.

[18] Cody Tinker, Kevin Millar, Alan Kaminsky, Michael Kurdziel, Marcin Lukowiak, and Stanisław Radziszowski, "Exploring the application of homomorphic encryption to a cross domain solution," in *MILCOM 2019 - 2019 IEEE Military Communications Conference (MILCOM)*, 2019, pp. 1–6. DOI: 10.1109/MILCOM47813.2019.9021015.

[19] Alan Kaminsky, Michael Kurdziel, Steve Farris, Marcin Łukowiak, and Stanisław Radziszowski, "Solving the cross domain problem with functional encryption," in *MILCOM 2021 - 2021 IEEE Military Communications Conference (MILCOM)*, 2021, pp. 49–54. DOI: 10.1109/MILCOM52596.2021.9652958.

[20] Kyrre Wahl Kongsgard, Nils Agne Nordbotten, Federico Mancini, Raymond Haakseth, and Paal E. Engelstad, "Data leakage prevention for secure cross-domain information exchange," *IEEE Communications Magazine*, vol. 55, no. 10, pp. 37–43, 2017. DOI: 10.1109/MCOM.2017.1700235.

[21] Ran Dubin, "Content disarm and reconstruction of pdf files," *IEEE Access*, vol. 11, pp. 38 399–38 416, 2023. DOI: 10.1109/ACCESS.2023.3267717.

[22] Ran Dubin, "Content disarm and reconstruction of rtf files a zero file trust methodology," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 1461–1472, 2023. DOI: 10.1109/TIFS.2023.3241480.

[23] Richard Stiennon, Russ B. Ernst, and Fredrik Forslund, "Sanitizing files," in *Net Zeros and Ones: How Data Erasure Promotes Sustainability, Privacy, and Security*. 2023, pp. 113–116.

[24] Ronald Mraz Jeffrey Menoher, "Secure cross border information sharing using one-way data transfer systems," [Online]. Available: https://www.academia.edu/65884746/Secure_Cross_Border_Information_Sharing_Using_One_way_Data_Transfer_Systems.

[25] Andika Candra Jaya, Cutifa Safitri, and Rila Mandala, "Sneakernet: A technological overview and improvement," in *2020 IEEE International Conference on Sustainable Engineering and Creative Computing (ICSECC)*, 2020, pp. 287–291. DOI: 10.1109/ICSECC51444.2020.9557509.

[26] "Rabbitmq," [Online]. Available: https://www.rabbitmq.com/.

[27] AUTEK, "Pst data diode," [Online]. Available: https://www.autek.es/en/cross-domain/pstdiode.

[28] "Opswat metadefender core," [Online]. Available: https://www.opswat.com/products/metadefender/core.

[29] Richard Stiennon, Russ B. Ernst, and Fredrik Forslund, "Enterprise data sanitization policy," in *Net Zeros and Ones: How Data Erasure Promotes Sustainability, Privacy, and Security*. 2023, pp. 143–157.

[30] Isha Yadav and Himanshu Gupta, "Designing data loss prevention system for the enhancement of data integrity in cyberspace," in *2023 5th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, 2023, pp. 1361–1365. DOI: 10.1109/ICAC3N60023.2023.10541823.

[31] Simon Liu and Rick Kuhn, "Data loss prevention," *IT Professional*, vol. 12, no. 2, pp. 10–13, 2010. DOI: 10.1109/MITP.2010.52.

[32] "Rabbitmq clustering," [Online]. Available: https://www.rabbitmq.com/docs/clustering.

[33] "Rabbitmq priority queue," [Online]. Available: https://www.rabbitmq.com/docs/priority.

# A. RabbitMQ Queue Creation Script

This appendix showcases the script to create the RabbitMQ queues and queue binding using python script.

```python
import pika

# RabbitMQ connection parameters
rabbitmq_host = 'localhost'   # RabbitMQ Server
port = '5672' # default port
virtualHost = 'VirtualHost'
exchange_name = 'exchangeName'
queue_name = 'queueName'
routing_key = 'routingKey'
userName = 'guest'
password = 'guest'

# Connect to RabbitMQ
credentials = pika.PlainCredentials(userName, password)
connParams = pika.ConnectionParameters(rabbitmq_host,
                                       port,
                                       virtualHost,
                                       credentials)

connection = pika.BlockingConnection(connParams)
channel = connection.channel()

# Declare an exchange
```

```python
channel.exchange_declare(
    exchange=exchange_name,
    exchange_type='topic',
    passive = False,
    auto_delete = True
    durable=True
)

# Declare a queue
channel.queue_declare(
    queue=queue_name,
    durable=True,
    excluside=False,
    auto_delete=False
)

# Bind the queue to the exchange with the routing key
channel.queue_bind(
    exchange=exchange_name,
    queue=queue_name,
    routing_key=routing_key
)

# Close the connection
connection.close()

```

## B. Method to Upload a file through Web API

```
using System;
using System.Net.Http;
using System.Threading.Tasks;

public class FileUploader
{
    public async Task<HttpResponseMessage>
     ↪ UploadFile(MultipartFormDataContent content, string
     ↪ endpointUrl)
    {
        using (HttpClient httpClient = new HttpClient())
        {
            try
            {
                HttpResponseMessage response = await
                 ↪ httpClient.PostAsync(endpointUrl, content);
                response.EnsureSuccessStatusCode(); // throws
                 ↪ exception if status code is not success
                return response;
            }
            catch (Exception ex)
            {
                // handle the exception
                throw new ApplicationException("File upload
                 ↪ failed", ex);
```

```
                    }
                }
            }
}

```

## C. RabbitMQ Queue Subscriber

```csharp
using System;
using System.Text;
using RabbitMQ.Client;
using RabbitMQ.Client.Events;

public class RabbitMQSubscriber
{
    public void SubscribeToQueue()
    {
        // Connection settings
        var factory = new ConnectionFactory()
        {
            HostName = "localhost",   // RabbitMQ host name
            UserName = "guest",       // username
            Password = "guest"        // password
            VirtualHost = "virtualHost" //Virtual Host Name
        };

        using (var connection = factory.CreateConnection())
        using (var channel = connection.CreateModel())
        {
            string queueName = "rabbitmqqueue"; // RabbitMQ queue
             ↪  name

            // Ensure the queue exists before trying to consume
            channel.QueueDeclare(queue: queueName,
                                 durable: true,
                                 exclusive: false,
                                 autoDelete: false,
                                 arguments: null);

            var consumer = new EventingBasicConsumer(channel);

            consumer.Received += (model, ea) =>
            {
                var body = ea.Body.ToArray();
                var message = Encoding.UTF8.GetString(body);

                //Business Logic goes here after message is
                 ↪   received

                //Acknowledge the message so it is removed from the
                 ↪   queue
                channel.BasicAck(deliveryTag: ea.DeliveryTag,
                 ↪   multiple: false);
            };

            // Start consuming
            channel.BasicConsume(queue: queueName,
                                 autoAck: false, // Use false for
                                  ↪   manual ack
                                 consumer: consumer);
        }
    }
}
```

## D. RabbitMQ Message Publisher

```
using System;
using System.Text;
using RabbitMQ.Client;

public class RabbitMQPublisher
{
    public void PublishMessage()
    {
        // RabbitMQ connection settings
        var factory = new ConnectionFactory()
        {
            HostName = "localhost",  // RabbitMQ host name
            UserName = "guest",      // username
            Password = "guest"       // password
            VirtualHost = "virtualHost" //Virtual Host Name
        };

        using (var connection = factory.CreateConnection())
        using (var channel = connection.CreateModel())
        {
            string exchangeName = "exchangeName";
            string routingKey = "routingKey";
            string message = "Test message through Exchange!";

            // Declare the exchange
            channel.ExchangeDeclare(exchange: exchangeName, type:
             ↪  "topic", durable: true);

            // Convert the message to a byte array
            var body = Encoding.UTF8.GetBytes(message);

            // Publish the message to the exchange with a routing
             ↪  key
            channel.BasicPublish(exchange: exchangeName,
                                 routingKey: routingKey,
                                 basicProperties: null,
                                 body: body);
        }
    }
}

```

# E. Encryption Method

```
using System;
using System.IO;
using nsoftware.SecureBlackbox;

public class Encryptor
{
    public void Encrypt(Stream inputStream, Stream outputStream)
    {
        try
        {
            // Load recipient certificate
            CertificateList recipients = getRecipientCertificate();

            // Set up the encryptor
            using (MessageEncryptor encryptor = new
             ↪  MessageEncryptor())
            {
                // Add recipient
                encryptor.EncryptionCertificates = recipients;

                // Set encryption algorithm
                encryptor.EncryptionAlgorithm = "AES256";

                // Assign input/output streams
                encryptor.InputStream = inputStream;
                encryptor.OutputStream = outputStream;

                // Perform encryption
                encryptor.Encrypt();
            }

        }
        catch (Exception ex)
        {
            throw;
        }
    }
}

```

# F. Signature Method

```csharp
using System;
using System.IO;
using nsoftware.SecureBlackbox;

public class SBBSigner
{
    public void Sign(Stream inputStream, Stream outputStream)
    {
        try
        {
            // Load the signing certificate with private key
            Certificate signerCert = getPrivateCertificate();

            // Set up the signer
            using(MessageSigner signer = new MessageSigner())
            {
                // Assign signer certificate
                signer.SigningCertificate = signerCert;

                // Choose the signature format
                signer.SignatureType =
                 ↪  MessageSignerSignatureTypes.stPKCS7Enveloping;
                signer.HashAlgorithm = "SHA256";

                // Assign input and output streams
                signer.InputStream = inputStream;
                signer.OutputStream = outputStream;

                // Perform signing
                signer.Sign();
            }
        }
        catch (Exception ex)
        {
            throw;
        }
    }
}

```

# G. Decryption Method

```
using System;
using System.IO;
using nsoftware.SecureBlackbox;

public class FileDecryptor
{
    public void Decrypt(Stream inputStream, Stream outputStream)
    {
        try
        {
            // Load recipient's certificate with private key
            Certificate recipientCerts = getPrivateCertificates();

            // Set up the decryptor
            using (MessageDecryptor decryptor = new
             ↪  MessageDecryptor())
            {
                // Assign recipient's certificates
                decryptor.Certificates = recipientCerts;

                // Assign input and output streams
                decryptor.InputStream = inputStream;
                decryptor.OutputStream = outputStream;

                // Perform decryption
                decryptor.Decrypt();
            }
        }
        catch (Exception ex)
        {
            throw;
        }
    }
}

```

# H. Verification Method

```csharp
using System;
using System.IO;
using nsoftware.SecureBlackbox;

public class SignatureVerifier
{
    public bool Verify(Stream inputStream, Stream outputStream)
    {
        signerInfo = null;
        //Retrieve public certs for signature verification
        CertificateList verifyCerts = getPublicCertificates();
        try
        {
            // Set up the verifier
            using (MessageVerifier verifier = new MessageVerifier())
            {
                verifier.KnownCertificates = verifyCerts;

                // Assign the signed input/output stream
                verifier.InputStream = inputStream;
                verifier.OutputStream = outputStream;

                // Perform verification
```

```
                    verifier.Verify();
                }

                // Check signature validation result
                return verifier.SignatureValidationResult ==
                 ↪  MessageVerifierSignatureValidationResults.svtValid;
            }
            catch (Exception ex)
            {
                return false;
            }
        }
    }

```