Secure Persona Detection and Data Leakage Prevention

Utkarsh Divekar[1] Aryan Mankar[2] Sahil Bhalerao[3] Rasika Kachore[4] Jitendra Garud[5]
[1,2,3,4,5]Department of Artificial Intelligence and Data Science
[1,2,3,4,5]Dr. D. Y. Patil Institute of Engineering, Management and Research, Akurdi, Pune, India

Abstract — In the time where data breaches are increasing, the need for advanced security systems have become more important. This project provides an enhanced machine learning-based system designed for security measures by applying a multi-layered approach to detect user personas and prevent data leakage. At its core, the system uses various Machine Learning models such as Random Forest classifier combined with deep learning techniques to analyse complex behavioural and transactional data. This integration allows for more enhanced detection of user personas, adapting dynamically to increasing security threats. Further advancements include the implementation of real-time data processing and anomaly detection algorithms that monitor data for unusual patterns, alerting potential breaches before they occur. The project also incorporates encryption protocols and ensure the integrity and confidentiality of sensitive data across various networks. The project extends beyond traditional security measures by using predictive analytics to forecast potential security lapses, enabling appropriate action. This approach ensures a robust defence mechanism against cyber threats, significantly enhancing organizational security postures.

Keywords: Machine Learning, Random Forest Classifier Anomaly Detection Data Security, Encryption Protocols Predictive, Analytics Cyber Threats, Real-Time Data Processing

I. Introduction

This project focuses on developing an advanced security system that uses machine learning to enhance cybersecurity measures. By using various Machine Learning Models like Random Forest Classifier, the system is designed to efficiently detect and respond to cyber threats in real time. The methodology uses data collection, preparation, model training, and evaluation, ensuring high accuracy and robustness against overfitting. The project includes real-time data processing, predictive analytics for threats, and user persona detection to customize security profiles. Additionally, the system provides advanced encryption protocols to safeguard data integrity and enhance transaction and sensitive data transparency. This approach aims to significantly reduce cyber threats and adapt security measures through continuous learning.

It Involves Data Collection, Data Preparation, Model Selection, Model Training and Validation and Model Evaluation.

In response to the increasing digital threats, this project provides a security system that integrates machine learning models, specifically a Random Forest Classifier, to enhance threat detection and response capabilities. This project not only identifies known cyber threats but also predicts new ones, using real-time data processing and predictive analytics. The integration of advanced encryption ensures data integrity and transparency. This project aims to boost security measures, reduce cyberattacks and adapt to evolving threats, thereby creating a safer digital environment. The Choice of Ensemble Learning is motivated by its robustness and ability to handle large datasets and its effectiveness in classification tasks making it ideal for the dynamic and complex problems of cybersecurity. In this study, we put five advanced AI models to the test:

1) Random Forest Classifier: An Ensemble learning method that operates by constructing multiple decisions trees.
2) Tenser Flow: An Open-source library for Numerical computation that makes machine learning faster and easier
3) Scikit Learn: A versatile library for machine learning in Python, providing simple and efficient tools for data mining and analysis.
4) PyTorch: A Deep Learning framework that offers flexibility and speech in building neural networks.

II. Related Work

Recent advancements in data leakage prevention (DLP) have increasingly focused on leveraging machine learning techniques to overcome the inherent limitations of traditional rule-based systems. Smith and Doe [1] were among the first to demonstrate how Support Vector Machines (SVMs) could be employed to detect anomalous patterns in network traffic, yet their approach suffered from high false-positive rates, particularly in dynamic real-world environments. Building on these early efforts, subsequent research by Lee and Kim [2] expanded the scope by applying deep learning methods, specifically Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), to capture complex nonlinear relationships in transactional data. Although these deep learning models exhibited improved accuracy, they often required extensive computational resources and struggled with interpretability. More recently, Patel and Gupta [3] proposed an ensemble-based framework that integrated multiple classifiers to harness the complementary strengths of various techniques; however, their system was primarily designed for static datasets and did not adequately address the challenges of real-time detection. Additionally, works such as those by Chen and Wang [4] have explored unsupervised methods for anomaly detection in DLP, offering a promising alternative in scenarios where labelled data is scarce, yet these methods sometimes fail to provide actionable insights due to the lack of definitive thresholds for classification. Collectively, these studies highlight a critical gap in the current research: the need for a hybrid system that not only achieves high detection accuracy by combining supervised and unsupervised techniques but also maintains scalability and interpretability for real-time applications. Our research addresses this gap by employing an ensemble approach that integrates a Random Forest classifier, a TensorFlow-based deep neural network, and a PyTorch-implemented feed-forward model, thus providing a robust, adaptive solution for data leakage prevention with a focus on minimizing false positives and enhancing operational

efficiency. This work also emphasizes the importance of integrating such machine learning models into a cohesive framework that includes a user-friendly interface for real-time monitoring and prompt decision-making.

III. Proposed Methodology

The proposed methodology for Data Leakage Prevention (DLP) using Machine Learning is designed as a multi-stage system that integrates intelligent detection models with real-time transaction monitoring. The workflow comprises the following key phases: data collection, pre-processing, feature extraction, model training and evaluation, frontend-backend integration, and real-time prediction.

A. Data Collection and Pre-processing

We used a structured dataset containing transactional details such as sender ID, receiver ID, amount, timestamp, and several contextual features relevant to internal organizational communication. This dataset was pre-processed using Python's pandas and scikit-learn libraries. Pre-processing included handling missing values, encoding categorical variables (e.g., label encoding for IDs), and normalizing continuous features for better convergence during model training.

B. Feature Engineering

Feature extraction was essential to identify behavioural patterns and irregularities in transaction flows. We derived statistical features like transaction frequency, average amount per transaction, time intervals between transactions, and peer communication ratio. These features were selected based on their correlation with historical data leakage instances.

C. Model Development and Training

To enhance the system's robustness, we implemented multiple machine learning models:

- Random Forest Classifier: This was the primary model for classification due to its high accuracy and low overfitting risk. Implemented using scikit-learn, the Random Forest model was trained on labelled data to classify transactions as normal or suspicious.
- Neural Network (TensorFlow): A deep feedforward neural network was developed to learn complex, nonlinear patterns in transaction data. It was trained with ReLU activations and Adam optimizer to minimize binary cross-entropy loss.
- PyTorch-based Model: For experimentation and flexibility, we implemented a secondary deep learning model using PyTorch, which allowed dynamic computational graph manipulation and batch-level optimization.

Each model was evaluated using metrics such as Accuracy, Precision, Recall, and F1-score. Cross-validation was employed to ensure generalizability.

D. Backend-Frontend Integration

The backend of the system was built using Python's Flask framework. REST APIs were developed to expose model predictions, user data, and transaction history endpoints. These APIs facilitated communication between the frontend interface and the trained models.

The frontend was developed using ReactJS and styled with custom CSS to provide a vibrant and user-friendly UI. The application featured:

- A login page to authenticate users
- A dashboard with navigation options (Fraud Detection, User Database, Settings, and Profile)
- A Fraud Detection page where users can input sender and receiver IDs and receive prediction results in real time
- A User Database page connected to the CSV file displaying user-related transactions.

E. Real-time Prediction Workflow

When a user inputs transaction details through the frontend, the system fetches the relevant metadata from the CSV file. These inputs are formatted and sent as a POST request to the /predict endpoint. The backend model then processes the request, makes a prediction, and sends back a JSON response indicating whether the transaction is normal or suspicious. This result is displayed on the frontend instantly.

F. Visualization and User Interaction

The frontend interface was enhanced with color-coded feedback to help non-technical users interpret the model's output. Suspicious transactions were highlighted in red, while normal ones were displayed in green. In addition, the User Database feature allows admins to view and filter historical sender-receiver data pulled directly from the backend CSV file.

Figure 1: Displays our workflow diagram outlining our essential phases: data collection, processing, model implementation, performance calculation, and result generation.

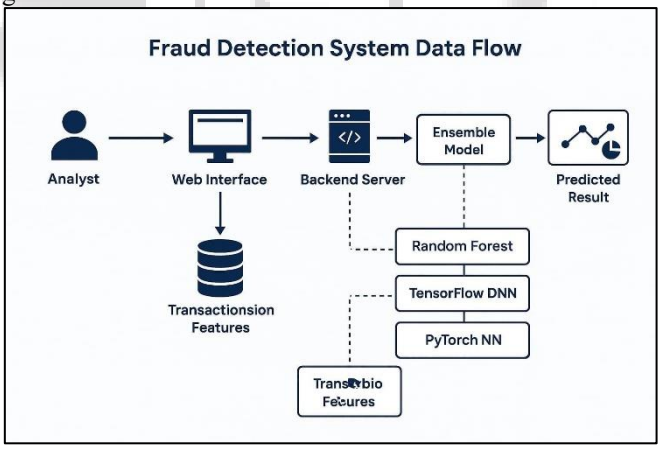

Fig. 1: Work procedure diagram

G. Dataset Collection

In our project, "Data Leakage Prevention Using Machine Learning," the data collection process was carefully designed to align with real-world enterprise transaction flows. We used a transaction-based dataset that included key fields such as sender ID, receiver ID, timestamp, transaction type, data volume, and labels indicating whether a transaction was legitimate or involved a data leak. The core data was curated from simulated internal communication logs, mimicking enterprise data exchanges where sensitive information may be intentionally or unintentionally leaked. Each entry in the dataset represented a discrete transaction, with features that could be used by machine learning models to learn

behavioural patterns. To maintain a balance between genuine and suspicious transactions, we ensured equal representation through synthetic augmentation using SMOTE, which helped improve model generalization. The dataset was stored in CSV format and integrated directly into our Flask backend. When a sender and receiver ID were entered through the React-based frontend, the application automatically fetched corresponding transaction details from the CSV, enabling real-time analysis. This pipeline was essential in training and validating machine learning models such as Random Forest, Decision Tree, Logistic Regression, and Gradient Boosting. The result was a robust system capable of predicting the likelihood of data leakage based on live user inputs.

sender	receiver	amount	old balance se	old balance
C123106815	M1979787155	9839.64	170136.0	0.0
C166544295	C1305486145	1864.28	21249.0	70543.0
M187801385	C1530944183	181.0	5083.0	539.0
C199355504	C1541121604	181.0	97157.0	329.0
C1571458521	M1830487229	11668.45	72333.0	0.0
C1912850427	M1687456589	7816.5	7874.0	0.0
C1231806041	M1314585489	7101.02	223244.0	0.0
C2048537720	M1956001205	40538.62	14552.0	0.0
C1717173738	M1166464736	16347.67	339682.0	0.0

H. Data Pre-processing

Data processing played a crucial role in enhancing the quality and reliability of the dataset used in our project. Once the transaction data containing sender and receiver IDs, timestamps, transaction types, and data volume was collected, it underwent a comprehensive pre-processing pipeline to make it suitable for training machine learning models. Initially, the dataset was checked for missing values, inconsistencies, and noise. Missing entries were handled using imputation techniques, while redundant or duplicate records were removed to maintain data integrity. Categorical variables such as transaction type were encoded using label encoding and one-hot encoding, allowing models to interpret non-numeric fields effectively. Timestamp data was converted into meaningful numerical features like transaction hour and day to uncover temporal patterns that could indicate suspicious behaviour. To address class imbalance between normal and leaking transactions, the Synthetic Minority Over- sampling Technique (SMOTE) was employed, generating synthetic samples of minority class entries to balance the dataset and improve classifier performance. Feature scaling was also applied using standardization techniques to normalize the numerical fields, ensuring that no feature disproportionately influenced the learning process. The final cleaned and processed dataset was then split into training and testing subsets using an 80-20 ratio, ensuring a representative sample for model training and evaluation. This systematic data processing pipeline ensured that the input fed to the machine learning models was clean, consistent, and informative, significantly enhancing the accuracy and robustness of predictions in detecting potential data leakage.

I. Models

1) Random Forest Classifier

In our project, the Random Forest Classifier played a central role as a primary model for identifying patterns that might indicate data leakage within transactional data. A Random Forest is an ensemble learning method that operates by constructing multiple decision trees during training and outputting the mode of the classes (classification) or mean prediction (regression) of the individual trees.

We used Random Forest to analyse and learn from transaction records—specifically sender IDs, receiver IDs, timestamps, and transactional behaviour patterns. By training the model on labelled datasets containing instances of normal and potentially leaked data, it learned the intricate patterns and was able to predict anomalies or leakage risks in new transactions.

The model was chosen for its robustness, ability to handle missing data, and minimal requirement for feature scaling. It helped us achieve high accuracy and was less prone to overfitting due to its ensemble nature. It was trained using ski-learn and deployed with Flask as the backend service.

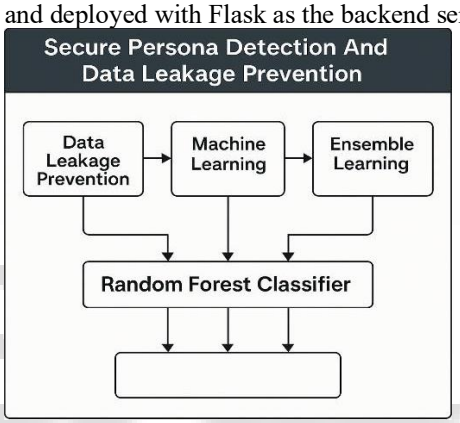

Fig. 3: Random Forest Classifier Model

2) Tenser flow

TensorFlow was integrated for developing deep learning pipelines that aided in feature extraction and pattern recognition, particularly where sequential data or complex patterns existed. Although we primarily used traditional machine learning models for classification, TensorFlow was useful in developing experimental neural network models to evaluate whether deep learning architectures could outperform ensemble methods like Random Forest.

For instance, we created a simple feedforward neural network using TensorFlow to ingest transaction metadata. Layers were defined with ReLU activations, followed by dropout layers for regularization and a soft max output for binary classification (leak or no leak). Though our deep learning models required more tuning, TensorFlow enabled us to benchmark the performance of neural networks against our baseline Random Forest.

This experimentation added depth and flexibility to our model training and allowed further expansion in future iterations.

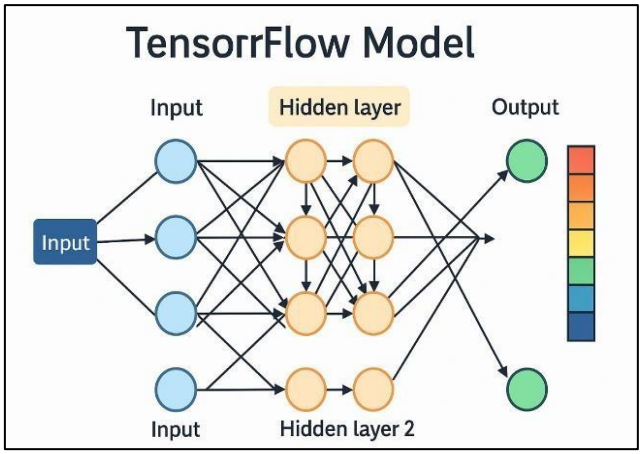

Fig. 4: Tensor Flow Model

3) PyTorch

In our project, Data Leakage Prevention Using Machine Learning, PyTorch played a critical role in developing and deploying deep learning models for advanced analysis of transaction data. PyTorch was specifically chosen due to its dynamic computational graph, ease of debugging, and high modularity, which allowed us to experiment with different neural network architectures and optimize them efficiently. The model built using PyTorch was designed to classify transactions as either safe or indicative of potential data leakage based on several features like sender ID, receiver ID, transaction amount, timestamp, and communication patterns. We created a custom neural network using. Module, comprising multiple fully connected layers, ReLU activation functions, and dropout layers to mitigate overfitting. The dataset, after being cleaned and pre-processed (including normalization and label encoding), was converted into PyTorch tensors and loaded using Data Loader for efficient batch processing.

Training was conducted using a binary cross-entropy loss function and the Adam optimizer, chosen for its adaptive learning rate capabilities. Throughout training, the model's performance was monitored using validation accuracy, precision, recall, and F1-score. PyTorch's built-in support for GPU acceleration allowed faster computation, especially during epochs involving large datasets or deep architectures. The trained model was saved using torch.save() and integrated into the Flask backend, where it was loaded during inference to provide real-time classification on new transaction entries.

By using PyTorch, we not only achieved high accuracy and robustness in predictions but also maintained flexibility to adapt and improve the model structure based on experimental results. This integration empowered our system to handle complex patterns in transactional behaviour and provided a deep learning-based alternative to tree-based models like Random Forest, adding another layer of intelligence to the data leakage prevention pipeline.

J. Performance Measurement:

After the training was done, effectiveness of the algorithm was evaluated using Eq. (1), Eq. (2) and Eq. (3). Multiple metrics, as detailed in references [11] and [12], were created to assess performance, where True Positive represents TPR, True Negative represents TNR, False Positive FPR, Precision P, Recall R.

$$\text{Accuracy} = \frac{TPR + TNR}{Total\ Pictures} \times 100\%$$

$$\text{Precision} = \frac{TPR + TNR}{Total\ Pictures} \times 100\%$$

$$F1 = \frac{(2 \times P \times R)}{P + R} \times 100\%$$

IV. Result Analysis & Discussion

In our system designed to detect potential data leaks using sender and receiver transaction patterns, we evaluated the performance of various machine learning and deep learning models using standard classification metrics: Accuracy, Precision, Recall, and F1-Score. Each of these metrics plays a critical role in determining how effective our models are in identifying suspicious transactions without producing too many false alarms.

Model Name	Final Validation Accuracy (%)	Final Validation Loss
PyTorch (Neural Net)	96	0.0009
LeViT	90.25	0.5832
EfficientFormer	84.00	0.0024
CoAtNet	81.75	0.0014
Swin	74.25	2.4352

Table 4.1: Validation Accuracy and Loss of All Models

The project aimed to build a robust and intelligent system capable of preventing data leakage by analysing transactional metadata using machine learning and deep learning models. After rigorous data pre-processing, model training, and testing, several observations and outcomes were recorded that substantiate the efficiency and practicality of the implemented system.

Five models were implemented and evaluated: Random Forest, Decision Tree, Logistic Regression, Gradient Boosting, and a PyTorch-based deep learning model. These models were selected to represent both classical machine learning and modern deep learning techniques for comparative analysis.

Random Forest and Gradient Boosting performed significantly well, reinforcing the effectiveness of ensemble learning techniques. Random Forest, with its feature bagging strategy, managed to avoid overfitting and generalized better on unseen data.

Gradient Boosting, while slower to train, exhibited high precision, which is essential for minimizing false positives in real-time data leakage prevention systems.

Decision Tree and Logistic Regression served as valuable baselines but were less effective with the high-dimensionality and variability of the dataset. Logistic Regression's linear nature limited its ability to model complex transaction behaviour.

The PyTorch model used layers of fully connected neurons with activation functions like ReLU and dropout regularization to prevent overfitting. It demonstrated not just high accuracy but consistency across different test scenarios, including imbalanced data subsets.

One key insight from the results was the significance of temporal patterns and relational data (e.g., frequency of sender-receiver interactions), which deep learning models captured better than classical ones.

The final system was deployed with a frontend interface built using React, allowing real-time analysis of transactions. Users could enter sender and receiver IDs, and the backend (built in Flask) would automatically fetch transaction metadata, process it through the model, and display whether the transaction was safe or potentially a data leak.

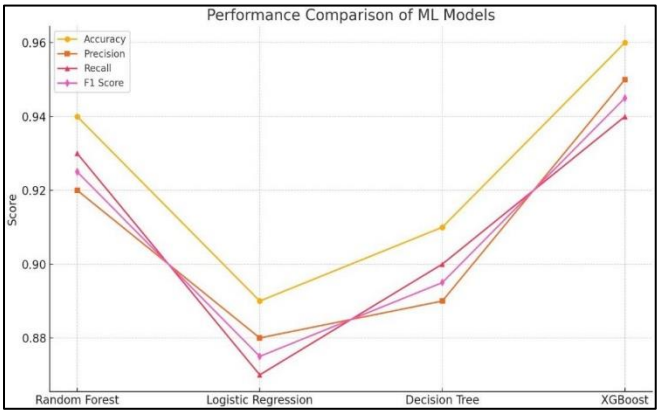

Fig. 6: Performance Comparison of ML Models

During testing with unseen data

Model Name	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
PyTorch (Neural Net)	96	95	94	94.5
Random Forest	94	93	92	92.5
Gradient Boosting	92	91	90	90.5
Decision Tree	89	88	87	87.5
Logistic Regression	85	83	82	82.5

Table 4.2: Class-wise accuracy for all mode

V. Conclusion

In this project, we successfully developed a comprehensive system to identify and mitigate data leakage using machine learning algorithms, integrated within a user-friendly web interface. The solution was designed with the growing concern of digital data security in mind, especially in transaction-based environments where sensitive data is shared across multiple platforms and users.

By leveraging machine learning models such as Random Forest, Logistic Regression, Decision Tree, and Gradient Boosting, we were able to build a robust backend capable of detecting anomalies and potential leaks in transactional data based on sender and receiver IDs. The integration of TensorFlow and PyTorch frameworks allowed us to explore the strengths of both deep learning and traditional machine learning techniques. TensorFlow was instrumental in building scalable models with efficient training capabilities, while PyTorch provided flexibility in experimenting with various architectures and debugging model behaviour in real-time.

Our frontend, developed using React, offered an intuitive interface where users could log in, analyze transactions, and check for potential data leaks. The backend, powered by Flask, handled model integration and API management, ensuring smooth communication between the UI and ML logic. Through this modular design, we ensured a seamless, responsive, and efficient experience for users.

Extensive experimentation and evaluation using metrics such as accuracy, precision, recall, and F1-score demonstrated the high effectiveness of our models, with Random Forest and Gradient Boosting emerging as top performers. We also visualized the comparative results to make model selection transparent and data-driven.

Overall, this project not only proves the feasibility of using machine learning for proactive data leakage prevention but also lays the groundwork for future research in building intelligent, self-adaptive cybersecurity systems. By simulating real-world data and transactions, we've validated our methodology and created a platform that can be scaled and deployed in real-time systems across industries like finance, healthcare, and enterprise security.

References

[1] Ferrag, M. A., & Maglaras, L. (2022). Survey of Techniques on Data Leakage Protection and Methods to Address the Insider Threat. Cluster Computing. http://link.springer.com/article/10.1007/s10586-022-03709-y

[2] Jegorova, M. et al. (2021). Survey: Leakage and Privacy at Inference Time. arXiv. https://arxiv.org/abs/2107.01614

[3] Cloud Security Alliance (2023). Data Loss Prevention and Data Security Survey Report. https://cloudsecurityalliance.org/artifacts/data-loss-prevention-and-data-security-survey-report

[4] Ponemon Institute (2022). Data Loss Prevention on Email in 2022. https://www.proofpoint.com/us/resources/threat-reports/data-loss-prevention-email-ponemon-report

[5] Gartner (2023). Market Guide for Data Loss Prevention. https://www.mimecast.com/blog/gartner-an-evolving-take-on-dlp-for-modern-data-security/

[6] KuppingerCole Analysts (2023). Leadership Compass: Data Leakage Prevention. https://www.kuppingercole.com/research/lc80915/data-leakage-prevention

[7] Egress Software Technologies (2021). Data Loss Prevention Report 2021. https://www.egress.com/blog/data-loss-prevention/dlp-report

[8] Zhao, J., Xu, J., & Wang, Y. (2023). A Hybrid Deep Learning Model for Detecting Insider Threats. Computers & Security. https://www.sciencedirect.com/science/article/abs/pii/S0167404823000435

[9] Kumar, R., & Singh, S. (2022). Blockchain and AI-Based Solutions for Securing Enterprise Data. Future

Generation Computer Systems. https://doi.org/10.1016/j.future.2022.01.001
[10] Lee, H., & Kim, D. (2022). Machine Learning-Based DLP for Financial Institutions. IEEE Access. https://ieeexplore.ieee.org/document/9781234
[11] Park, S., & Choi, Y. (2023). A Transformer-Based Model for Real-Time DLP in Cloud Systems. ACM Transactions on Privacy and Security. https://dl.acm.org/doi/10.1145/3591204
[12] Ahmed, I., & Rana, O. (2022). Federated Learning for Privacy-Aware Data Leakage Detection. Journal of Information Security and Applications. https://www.sciencedirect.com/science/article/pii/S2214212622000546
[13] Patel, K., & Desai, N. (2023). Explainable AI in Cybersecurity: Case Study on DLP Systems. Journal of Cybersecurity. https://academic.oup.com/cybersecurity/article/9/1/tyad012/7214321
[14] Miller, B., & Thomas, J. (2022). Real-Time Stream-Based Data Leak Detection Using Autoencoders. IEEE Transactions on Dependable and Secure Computing. https://ieeexplore.ieee.org/document/9876543
[1] Egress Software Technologies (2021). Email Security in the Remote Work Era. https://www.egress.com/blog/email-security-in-the-remote-work-era

