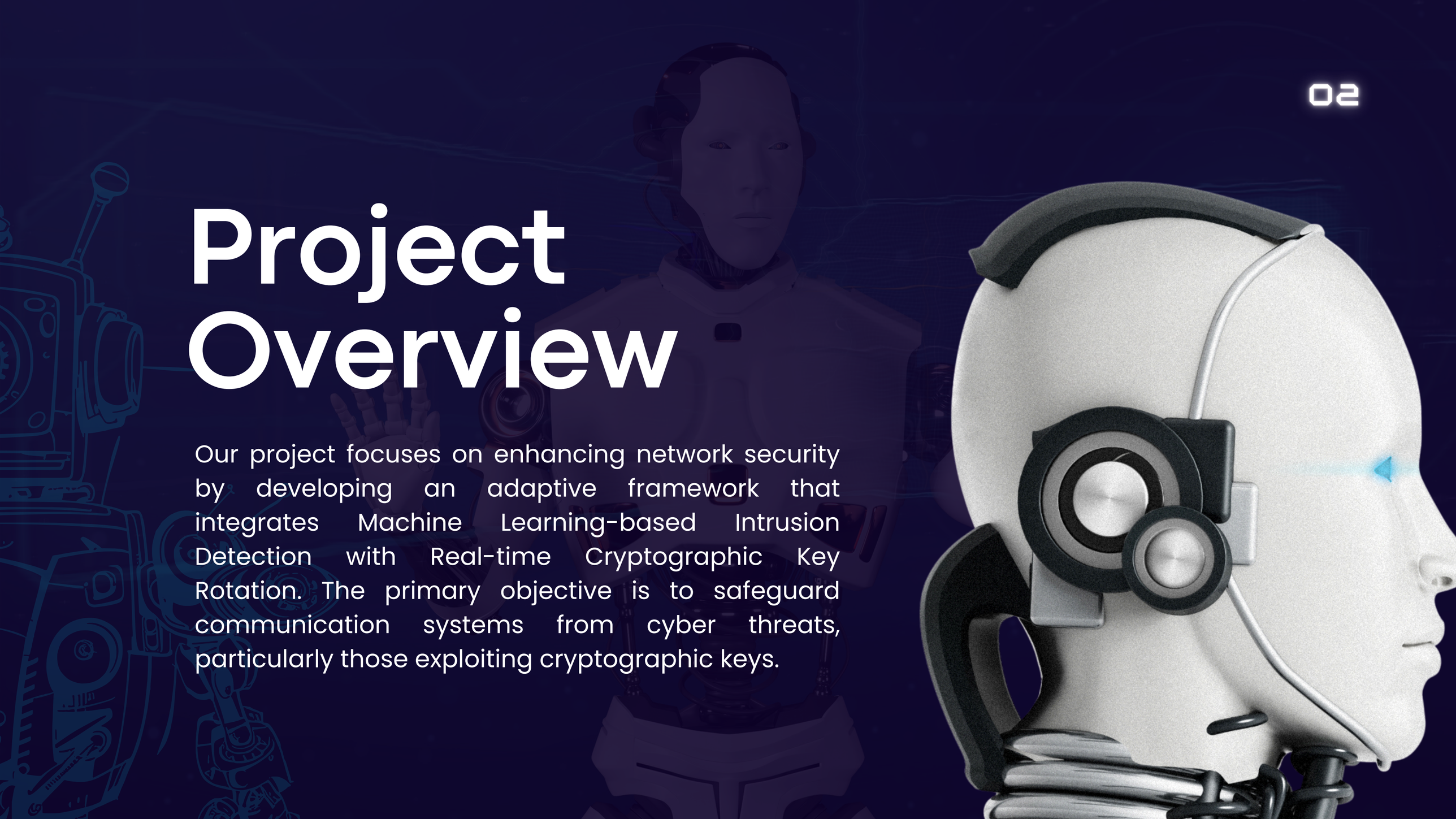
PROJECT REVIEW 2

# ADAPTIVE CRYPTOGRAPHIC KEY MANAGEMENT USING MACHINE LEARNING BASED INTRUSION DETECTION

Arshia -1409
Nidhi -1429
Haseena - 1449

# Project Overview

Our project focuses on enhancing network security by developing an adaptive framework that integrates Machine Learning-based Intrusion Detection with Real-time Cryptographic Key Rotation. The primary objective is to safeguard communication systems from cyber threats, particularly those exploiting cryptographic keys.



# Section in Focus

## Machine Learning

Data cleaning, EDA , Encoding, Model Evaluation

## Cryptography

Comparioson , Evaluation, Testing of Potential cryptography algorithm.

## Front End

Structural Prototyping of required frontend Interface

# Data Collection and Preprocessing

05

## Data Cleaning & Imputation

- Loaded dataset  and removed duplicate entries.
- Handled missing values (NaNs) using two strategies:
- Numerical Features: Mode imputation (e.g., for ip_reputation_score).
- Categorical Features: Replaced with a distinct "Unknown" category.

## Feature Engineering & Transformation

- Dropped the non-predictive session_id column.
- Converted categorical features (protocol_type, encryption_used) into numerical format using One-Hot Encoding.
- Used drop_first=True during encoding to mitigate multicollinearity.

## Scaling & Data Splitting

- Applied StandardScaler to all numerical features to normalize their range on training data
- The final dataset was split 80/20 using a stratified split on the attack_detected target to ensure identical class distribution in both sets.

```
df.head(5)
```

| | session_id | network_packet_size | protocol_type | login_attempts | session_duration | encryption_used | ip_reputation_score | failed_logins | browser_type | unusual_time_access | attack_detected |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | SID_00001 | 599 | TCP | 4 | 492.983263 | DES | 0.606818 | 1 | Edge | 0 | 1 |
| 1 | SID_00002 | 472 | TCP | 3 | 1557.996461 | DES | 0.301569 | 0 | Firefox | 0 | 0 |
| 2 | SID_00003 | 629 | TCP | 3 | 75.044262 | DES | 0.739164 | 2 | Chrome | 0 | 1 |
| 3 | SID_00004 | 804 | UDP | 4 | 601.248835 | DES | 0.123267 | 0 | Unknown | 0 | 1 |
| 4 | SID_00005 | 453 | TCP | 5 | 532.540888 | AES | 0.054874 | 1 | Firefox | 0 | 0 |

```
Class Distribution in Train Set:
0    0.55289
1    0.44711
Name: attack_detected, dtype: float64
Class Distribution in Test Set:
0    0.552935
1    0.447065
Name: attack_detected, dtype: float64
```

# Exploratory Data Analysis (EDA) & Key Insights

## Target Variable Analysis :

- Finding: The dataset is well-balanced (55% Normal, 45% Attack).
- Insight: We can train models directly without needing to fix class imbalance.

## Outlier Detection :

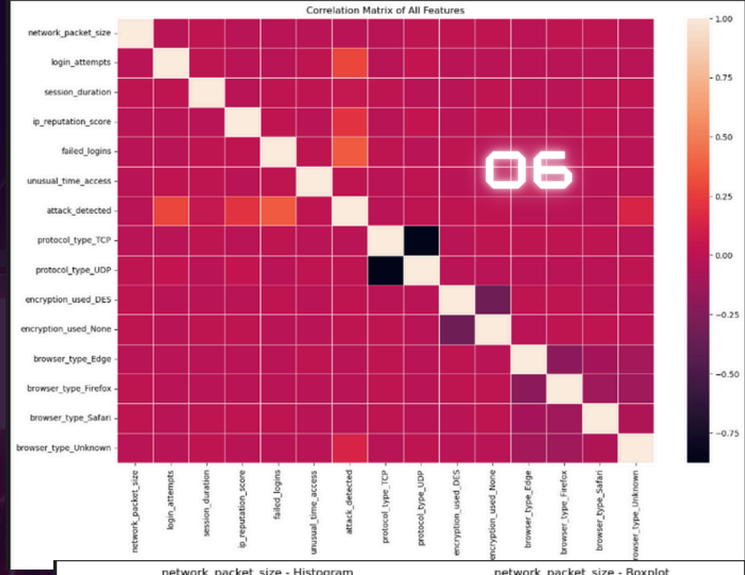
- Finding: Boxplots confirmed significant outliers in numerical features like session_duration and network_packet_size.
- Insight: This validates our use of median imputation and StandardScaler.

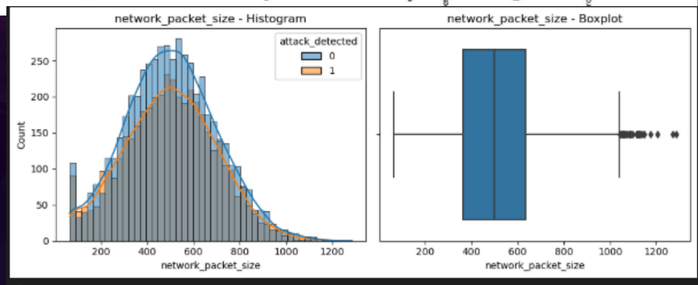
## Bivariate Analysis :

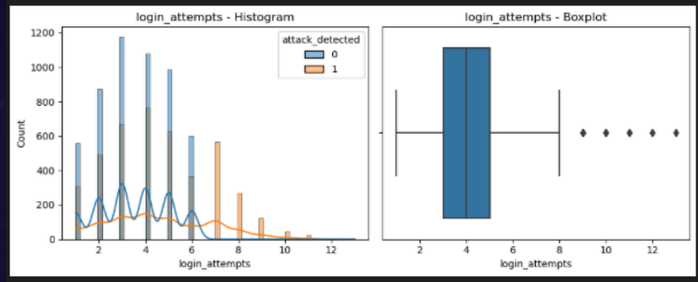
- Finding: Histograms showed clear distributional differences for features like failed_logins between "Attack" (1) and "Normal" (0) classes.
- Insight: This demonstrates a strong, separable signal that models can easily learn.

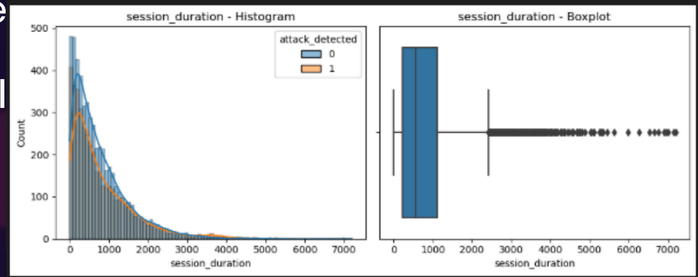
## Feature Correlation :

- Finding: The heatmap showed strong correlations between the target and features like failed_logins, ip_reputaion_score, etc.
- Insight: This confirms our engineered features are highly predictive and that the model have strong relationships to leverage.



# Evaluation Metrics

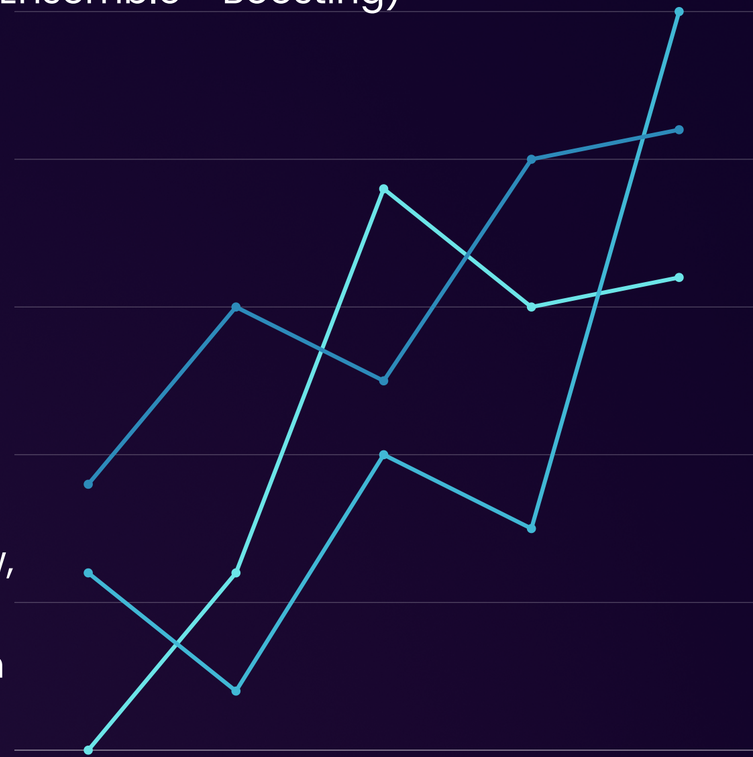
- Performance Metrics
- Accuracy: The percentage of all predictions (both Attack and Normal) that were correct.
- Precision: The percentage of flagged attacks that were actually attacks.
- Recall: The percentage of actual attacks that the model successfully caught.
- F1-Score: The balanced average of Precision and Recall; useful when class distribution is uneven.
- ROC-AUC: The model's overall ability to correctly distinguish between "Attack" and "Normal" classes.

Models Benchmarked We tested five different types of classifiers to compare their approaches:

- Logistic Regression (Linear Model)
- K-Nearest Neighbors (KNN) (Distance-Based)
- Support Vector Machine (SVM) (Margin-Based)
- Random Forest (Ensemble - Bagging)
- XGBoost (Ensemble - Boosting)

```
=== Model Performance Comparison ===

                     Accuracy  Precision  Recall  F1 Score     AUC  Train Time (s)  Predict Time (s)  Model Size (MB)
Logistic Regression    0.7290     0.7193  0.6460    0.6807 0.7876          0.0306            0.0020           0.0011
Random Forest          0.8863     0.9969  0.7479    0.8547 0.8800          1.7291            0.0492          16.4134
SVM                    0.8737     0.9665  0.7433    0.8403 0.8730          6.5412            0.4349           0.3899
KNN                    0.8014     0.8624  0.6612    0.7485 0.8356          0.0246            0.1107           0.9950
XGBoost                0.8821     0.9758  0.7550    0.8513 0.8813          2.0915            0.0055           0.2251
```



Operational Metrics

- Train Time (s): The total time in seconds required to build and train the model.
- Predict Time (s): The average time in seconds needed for the trained model to classify a new, single data point.
- Model Size (MB): The amount of memory (in megabytes) the final trained model occupies on disk or in memory.

# Cryptography



## Encryption

Keeps data hidden from unauthorized users

## Key Exchange

Ensures the key exchange is trusted and untampered

## Dynamic Key Cryptography

Maintains secure communication even during attacks

# Encryption Techniques

| Parameter | AES | ASCON | ChaCha20 | ECC |
|---|---|---|---|---|
| 1. Security Strength | Very high – considered an industry standard and resistant to brute- | High – approved by NIST for lightweight encryption and secure for | High – provides strong protection against modern cryptographic | Very high – based on strong mathematical security principles. |
| 2. Speed / Performance | Fast – optimized for both hardware and software implementations. | Very fast – lightweight operations ensure minimal computational | Very fast – ideal for real-time or streaming data encryption. | Moderate – involves complex mathematical computations leading |
| 3. Adaptability for Dynamic Key Rotation | Excellent – supports easy and frequent rekeying, making it suitable for ML-based adaptive systems. | Limited – lacks built-in mechanisms for dynamic key rotation. | Limited – requires additional logic to support rekeying mechanisms. | Excellent – allows frequent generation of new key pairs efficiently. |
| 4. ML-Based IDS Compatibility | Works seamlessly with machine learning models and is widely used in adaptive security frameworks. | Moderate – effective for IoT use but less responsive to ML-driven adaptations. | Moderate – not directly designed for ML-triggered rekeying or adaptive systems. | Excellent – frequently integrated into ML-assisted adaptive cryptographic architectures. |
| 5. Resource / Energy Efficiency | Moderate – provides a good balance between power use and performance. | Excellent – extremely lightweight and efficient for resource-limited environments. | Good – designed for speed and low power consumption. | Moderate – efficient but computationally demanding for smaller devices. |

# Key Exchange

| Parameter | Diffie–Hellman (DH) | Elliptic Curve Diffie–Hellman (ECDH) | Elliptic Curve Cryptography (ECC) |
|---|---|---|---|
| 1. Security Strength | Strong, based on discrete logarithm, but requires large keys | Very strong; elliptic curve math offers equal strength with smaller keys | Very strong; supports both encryption and key exchange |
| 2. Speed / Performance | Slow due to large prime number computations | Fast; small keys and efficient operations | Moderate; faster than RSA but slower than ECDH |
| 3. Adaptability for Dynamic Key Rotation | Limited; heavy key generation not suitable for frequent updates | Excellent; lightweight key generation supports frequent rekeying | Excellent; supports frequent key updates and digital signatures |
| 4. Compatibility with ML-Based IDS | Poor; unsuitable for real-time ML-triggered rekeying | Excellent; widely used in adaptive, ML-based, and IoT systems | Excellent; integrates well with adaptive cryptographic frameworks |
| 5. Resource / Energy Efficiency | High resource consumption due to complex operations | Very efficient; small keys and low CPU usage | Efficient; lightweight computations |
| 6. Implementation Complexity | Complex; involves large mathematical calculations | Moderate; well-supported in modern cryptographic libraries | Moderate; additional functionality but manageable |
| 7. Suitability for Adaptive Cryptographic Systems | Low; traditional and static | High; best suited for adaptive and dynamic key management | High; supports both encryption and key exchange in adaptive systems |

# Cryptography conclusion

05

## AES (Advanced Encryption Standard):

AES is employed to ensure fast and highly secure data encryption, maintaining the confidentiality of transmitted information. It protects sensitive data from unauthorized access during communication.

## ECDH (Elliptic Curve Diffie–Hellman):

ECDH is used for efficient and secure key exchange between communicating parties. It provides strong security with smaller key sizes, ensuring the integrity and confidentiality of the shared secret keys.

## Dynamic Key Cryptography:

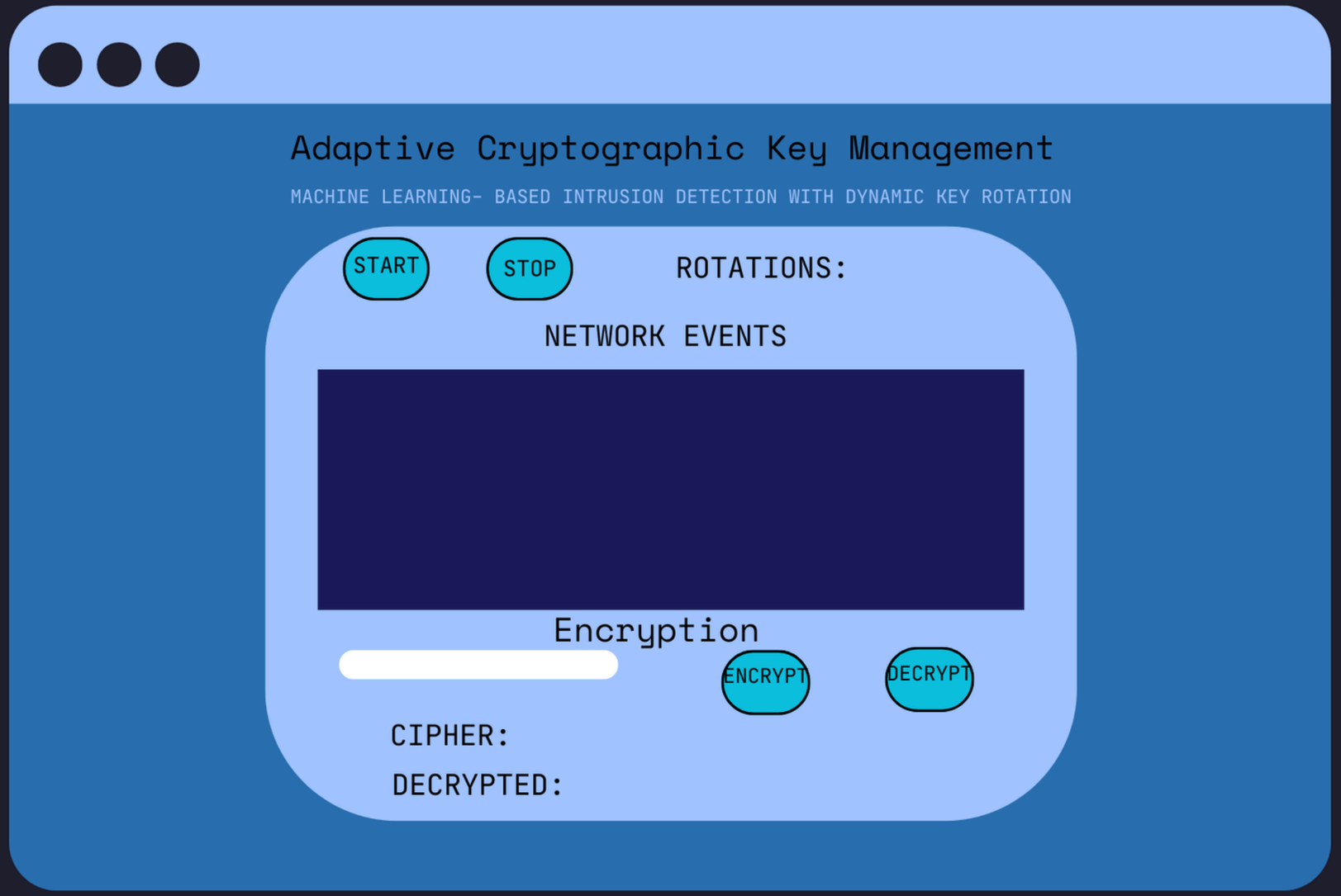
Dynamic Key Cryptography enables automatic key rotation whenever an intrusion or anomaly is detected by the Machine Learning-based Intrusion Detection System. This approach enhances system resilience by continuously adapting encryption keys to prevent potential attacks.

## Overall Integration:

The combination of AES, ECDH, and Dynamic Key Cryptography ensures confidentiality, integrity, and availability of data. It provides a robust, adaptive, and intelligent cryptographic framework that responds dynamically to evolving security threats.

# Design

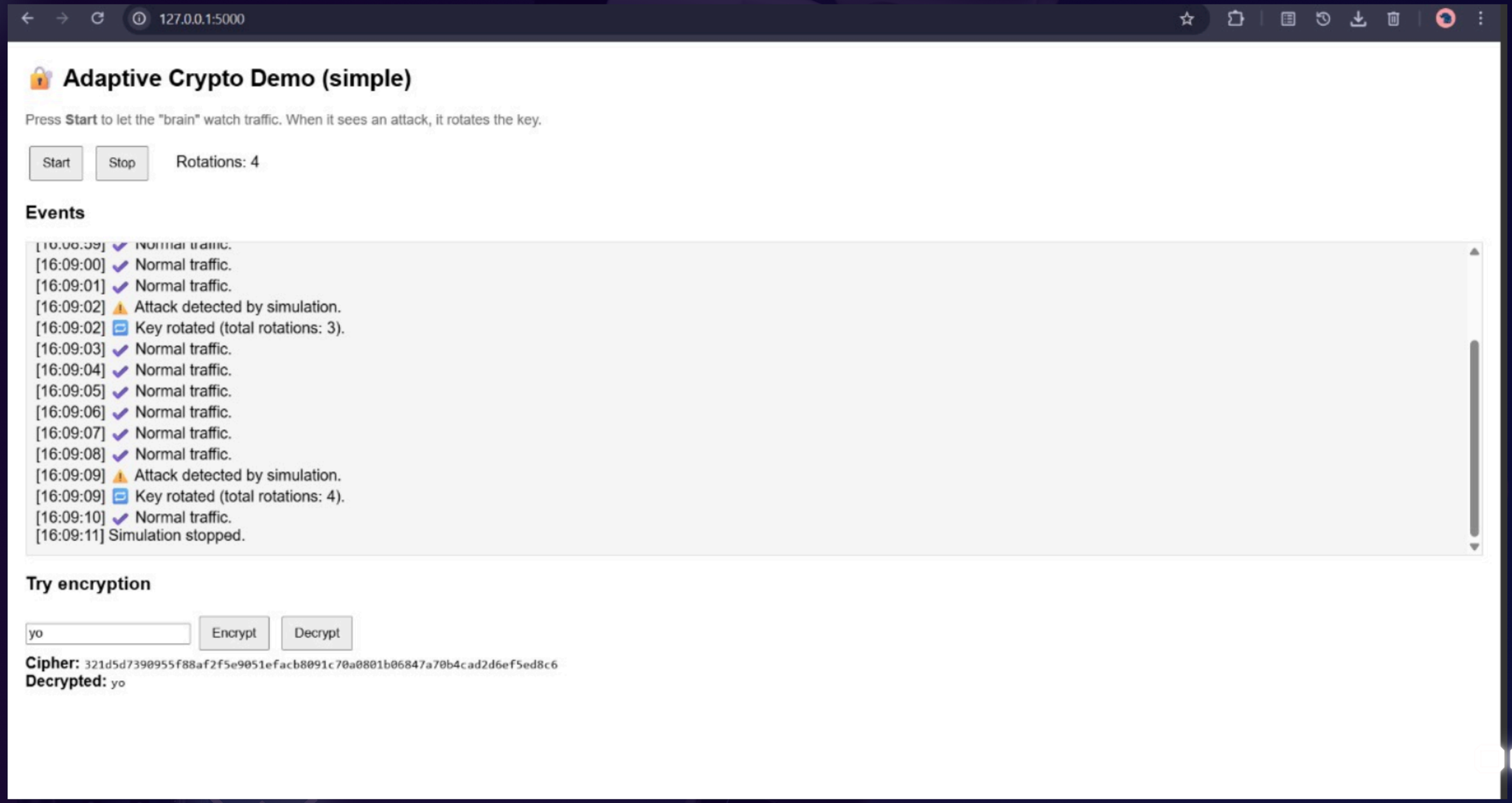



Adaptive Cryptographic Key Management
MACHINE LEARNING- BASED INTRUSION DETECTION WITH DYNAMIC KEY ROTATION
START
STOP
ROTATIONS:
NETWORK EVENTS
Encryption
ENCRYPT
DECRYPT
CIPHER:
DECRYPTED:

# HTML

- DOCTYPE & Head Section: Declares HTML5 document type. The <head> includes character encoding (UTF-8), a page title “Adaptive Crypto Demo”, and links an external CSS file (style.css) for styling.
- Header Content: Displays a main heading (<h2>) titled “Adaptive Cryptographic Key Management” with a lock emoji, and a subtext paragraph describing the project’s purpose — Machine Learning–based Intrusion Detection with Dynamic Key Rotation.
- Main Container (.container): Holds all interactive elements of the demo: Control Buttons: “Start” and “Stop” buttons to begin or stop key rotation, with a live counter (rotCount) showing the number of rotations.
- Network Events Section: Title “📡 Network Events” and a scrollable box (#events) that initially shows “No events yet. Press Start to begin.”
- Encryption Test Section: Title “🔐 Try Encryption” followed by an input field (#plain) for typing a message, and two buttons — Encrypt and Decrypt.
- Result Display: Two code blocks showing the generated cipher text (#cipher) and decrypted output (#decrypted).
- Script Section: Links the external JavaScript file (main.js) that handles interactive logic for buttons, encryption, and events.

# HTML


127.0.0.1:5000
Adaptive Crypto Demo (simple)
Press Start to let the "brain" watch traffic. When it sees an attack, it rotates the key.
Start
Stop
Rotations: 4
Events
[16:09:00] Normal traffic.
[16:09:01] Normal traffic.
[16:09:02] Attack detected by simulation.
[16:09:02] Key rotated (total rotations: 3).
[16:09:03] Normal traffic.
[16:09:04] Normal traffic.
[16:09:05] Normal traffic.
[16:09:06] Normal traffic.
[16:09:07] Normal traffic.
[16:09:08] Normal traffic.
[16:09:09] Attack detected by simulation.
[16:09:09] Key rotated (total rotations: 4).
[16:09:10] Normal traffic.
[16:09:11] Simulation stopped.
Try encryption
yo
Encrypt
Decrypt
Cipher: 321d5d7390955f88af2f5e9051efacb8091c70a0801b06847a70b4cad2d6ef5ed8c6
Decrypted: yo

# CSS

- Body: Set with a blue gradient background for a modern look. The text is white, centered, and uses the Poppins font. Flexbox layout ensures all elements are neatly aligned and responsive.
- Container: Styled with a transparent background and a blur effect to create a glassmorphism appearance. It has rounded corners, padding, and a soft shadow to make it stand out.
- Headings & Paragraphs: Headings (h2) are made larger with slight letter spacing, while paragraphs (p) use a lighter gray tone for better readability and visual hierarchy.
- Buttons: Designed in cyan color with rounded edges. They include hover effects that change the shade and slightly scale the button for interactivity.
- Events Section (#events): Given a dark, semi-transparent background with padding and rounded corners. It’s scrollable to neatly display logs or dynamic content.
- Input Fields: Styled with soft, rounded corners and no visible borders. Padding and spacing make them look clean and modern.
- Code Blocks: Highlighted with a dark background and yellowish text to distinguish code snippets from normal text.

# CSS


Adaptive Cryptographic Key Management
Machine Learning–based Intrusion Detection with Dynamic Key Rotation
Start
Stop
Rotations: 103
Network Events
[16:43:23] Normal traffic.
[16:43:24] Normal traffic.
[16:43:25] Normal traffic.
[16:43:26] Attack detected by simulation.
[16:43:26] Key rotated (total rotations: 103).
[16:43:27] Normal traffic.
[16:43:28] Normal traffic.
[16:43:29] Normal traffic.
[16:43:30] Normal traffic.
[16:43:31] Normal traffic.
[16:43:32] Simulation stopped.
Try Encryption
hello secret
Encrypt
Decrypt
Cipher:
Decrypted:

# Thank You!